그래프 탐색을 이용한 웹으로부터의 온톨로지 기반 규칙습득

박상언 [a], 이재규 [b], 강주영 [c]

[a b] 한국과학기술원 테크노경영대학원
서울 동대문 구 청량리 동 207-43
Tel: +82-2-958-3662, Fax: +82-2-960-2102, E-mail:{mascon, jklee}@kgsm.kaist.ac.kr

[c] 아주대학교 e비즈니스학부
수원시 영통구 원천동 산5번지
Tel: +82-31-219-2910, Fax: +82-31-219-1616, E-mail:jykang@ajou.ac.kr

Abstract

지능형 에이전트와 규칙기반 시스템을 이용해 보다 지능적인 웹 환경을 구축하고자 하는 노력이 시맨틱 웹의 발전과 함께 증가하고 있다. 이러한 에이전트와 규칙기반 시스템에 필요한 규칙들을 이미 많은 지식들이 산재해 있는 웹으로부터 습득할 수 있다면 보다 효율적으로 시스템을 구축하는 것이 가능하며, 이러한 응용시스템의 확장은 시맨틱 웹의 발전을 더욱 가속화하는 계기가 될 수 있을 것이다. XRML 방법론은 웹으로부터 규칙을 습득하기 위한 단계적 방법을 제시하고 있으며, 온톨로지를 이용함으로써 규칙의 구성요소들을 자동으로 추출할 수 있도록 지원한다. 그러나 추출된 규칙구성요소들을 조합하여 완전한 규칙을 만드는 과정이 규칙관리자의 수작업에 의존하고 있다. 본 연구는 온톨로지와 그래프 탐색을 사용함으로써 이 과정을 자동화하고자 하는 연구이다. 온톨로지에 있는 규칙의 일반적 패턴을 기반으로 하여 그래프 탐색을 이용해 규칙구성요소들을 조합함으로써 웹 페이지로부터 자동으로 규칙을 추출할 수 있다.

Keywords:

Rule Identification(규칙식별), Rule Acquisition(규칙습득), Ontology (온톨로지), Knowledge Acquisition(지식습득), OntoRule, Ontology Engineering(온톨로지공학), XRML, RuleML, XML

1. 서론

시맨틱 웹 방법론은 소프트웨어가 처리할 수 있는 형태로 정보를 표현하기 위하여 RDF(S)[4][15], OWL[22] 그리고 SWRL[11] 과 같은 표준언어들을 개발해 왔다. 그 결과, RDF와 같은 간단한 온톨로지 표현언어를 기반으로 한 논리로부터 다양한 종류의 논리들[8][9][20]까지 처리범위가 확장되었으며, 이상의 논리를 지원하기 위해 Jena[19], F-OWL[25], KAON[23], OntoBroker[7] 등과 같은 다양한 도구들이 개발되었다. 이와 같은 IT 기술의 발달은 소프트웨어로 하여금 웹을 보다 잘 이해하고 처리할 수 있도록 하였으며, 그로 인해 시맨틱 웹은 소프트웨어 에이전트가 사용자들에게 보다 복잡한 형태의 서비스를 제공할 수 있는 환경을 만들어가고 있다[3]. 따라서 앞으로 지능화된 서비스를 제공하는 소프트웨어 에이전트와 규칙기반 시스템들이 가까운 시일 내에 더욱 활발하게 제공될 수 있을 것으로 기대된다. 이러한 시스템들은 사실베이스 혹은 규칙베이스 형태로 온톨로지와 규칙들을 필요로 하며, 온톨로지를 추출하기 위한 다양한 방법론들이 제시됨에 따라 데이터베이스, 파일, 웹 페이지 등의 출처들로부터 온톨로지를 획득하기가 점점 쉬워지고 있다[1][6][10][24].

그러나 웹 페이지나 텍스트로부터 규칙을 습득하는 것은 온톨로지와 달리 쉽지 않다. 간단한 개념 수준의 온톨로지에 비해 규칙의 구조는 훨씬 복잡하며, 자연어로 표현된 규칙의 설명 역시 다양한 패턴을 갖고 있기 때문에 분석이 어렵기 때문이다. 과거 전문가 시스템에서의 경험으로 알 수 있듯이, 규칙의 부족은 소프트웨어 에이전트, 규칙기반 시스템과 같은 시맨틱 웹의 응용프로그램들의 활성화에 병목현상을 일으킬 수 있다. 따라서 본 논문의 목표는 이러한 병목현상을 해결하기 위하여 웹 페이지로부터 규칙을 자동으로 습득하는 방법론을 제시하고자 하는 것이다. 이와 같은 방법론을 통해 텍스트 및 웹 페이지로부터 규칙을 습득하는 새로운 관점을 제공함으로써, 본 연구가 지식병목현상을 해결하는데 기여할 수 있을 것으로 기대한다.

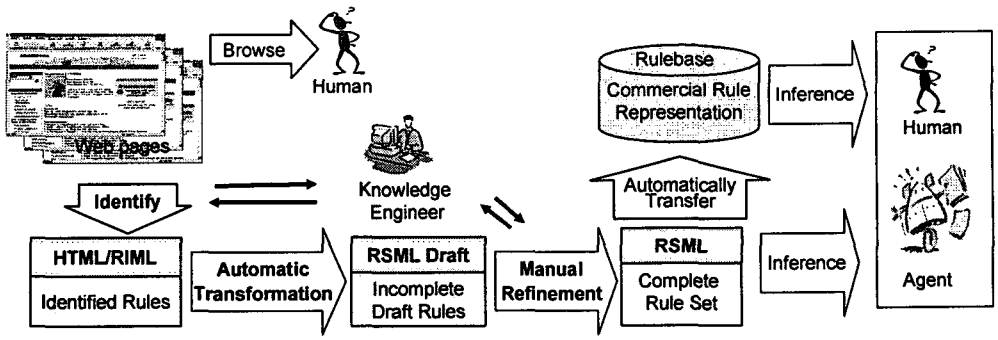

그림 1 – 온톨로지을 활용한 규칙습득의 절차

XRML(eXtensible Rule Markup Language) 방법론은 텍스트와 테이블로 이루어진 웹 페이지로부터 규칙을 습득하기 위한 프레임웍을 제공하고 있다 [13][14]. 다양한 규칙설명에 기인한 규칙구조의 복잡성으로 인해 자연어로 표현된 웹 페이지로부터 직접 규칙을 습득하는 것은 매우 어려운 것으로 알려져 있다 [21]. 규칙베이스에서 규칙들은 전형적으로 IF – THEN 구조를 갖고 있는 반면, 웹 페이지 상에서 이와 같은 형태로 설명되어 있는 경우는 거의 찾아볼 수 없다. 따라서 XRML 방법론은 웹으로부터의 규칙습득을 위한 단계적인 방법론을 그림 1과 같이 제시하고 있다. 첫째 단계는 웹 페이지로부터 변수 혹은 변수값과 같은 규칙구성요소들을 추출하여 ***RIML(Rule Identification Markup Language)*** 문서를 생성하는 것이다. 둘째 단계에서는 생성된 RIML 문서를 실행가능한 규칙의 형태로 자동변환하여 RSML 초안을 생성한다. 마지막 단계에서 지식관리자는 아직 불완전한 규칙을 수정하여 완전한 규칙집합을 만든다 [13].

이와 같이 규칙 습득과정은 규칙 구성요소의 식별에 좌우되는데, 일단 규칙 식별이 완료되면 이후는 대부분 자동화되어 있기 때문에 어렵지 않게 추론 가능한 규칙을 얻을 수 있다. 실제로 Kang과 Lee [13]의 논문에서 첫 단계인 규칙구성요소의 식별이 규칙생성을 상당부분 자동화하는 기반을 제시하고 있음을 규칙식별의 성능에 대한 실험을 통해 입증하였다.

그러나 다른 단계들이 상당부분 자동화되어 있는 것에 비해, 규칙구성요소를 식별하는 작업은 대부분 지식관리자의 수작업에 의존하고 있다. 이를 보완하기 위하여 후속 연구[17]에서는 규칙 온톨로지를 이용하여 규칙식별과정을 자동화하고자 노력하였다. 자동화된 규칙식별의 기본 아이디어는 동일한 도메인의 유사한 시스템에서 사용된 규칙베이스가 있는 경우 이 규칙베이스에 있는 규칙들을 일반화하여 온톨로지를 만들고, 이 온톨로지를 이용하여 변수나 변수값과 같은 규칙구성요소들을 추출하는 것이다. 기존의 시스템이 현재 구축하려는 시스템과 유사하다는 가정하에 이 방법은 성공적인 결과를 보였다. 또한 온톨로지가 반복된 규칙습득으로 인해 계속해서 축적되는 경우, 시스템의 성능은 더욱 나아질 것으로 기대된다. 그러나 추출된 규칙구성요소들을 결합하여 규칙을 생성하는 단계는 아직도 많은 부분이 규칙관리자의 수작업에 의존하고 있다.

따라서 본 연구에서는 자동으로 추출된 변수와 변수값들로부터 온톨로지를 이용하여 최적우선탐색(Best-First Search)을 통해 규칙을 자동으로 생성하는 방법론을 제안하고자 한다. 이를 위하여 최적우선탐색의 복잡성을 줄이기 위한 방법과 현재의 규칙에 적절한 변수를 선택하기 위한 평가함수를 제시하고자 한다. 다음으로는 이를 기반으로 하여 규칙을 생성하는 최적우선탐색 알고리즘을 설계하고자 한다. 온톨로지와 제안된 탐색 방법을 통해, 유사한 시스템과 규칙베이스가 존재하는 경우 규칙을 습득하는 대부분의 과정이 자동화될 수 있을 것으로 기대된다. 그림 2는 온톨로지를 이용한 전반적인 규칙구성요소 식별 과정을 보여주고 있다. 규칙 온톨로지는 ***OntoRule***로, 규칙식별 지원도구는 ***OntoXRML***로 각각 명명하였다. 그림 2에서 둘째 단계인 ***'Compose Rules'***가 본 연구의 핵심주제라 할 수 있다.

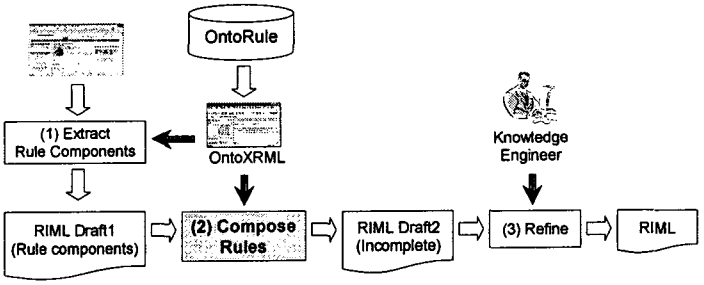

그림 2 – 온톨로지를 활용한 규칙 식별 절차

이 논문은 다음과 같이 구성되어 있다. 2장은 온톨로지를 이용한 규칙습득의 일반적인 방법론을 간략히 설명하고 있다. 3장에서는 규칙 구성에 있어 최적우선탐색을 사용하는 것과 관련된 논점들을 논의한다. 그리고 4장에서는 최적우선탐색을 이용한 규칙생성 알고리즘을 설명한다. 5장에서 간단한 실험을 통해 알고리즘을 평가하고 마지막 6장에서는 본 논문의 기여와 향후 연구에 대해 서술한다.

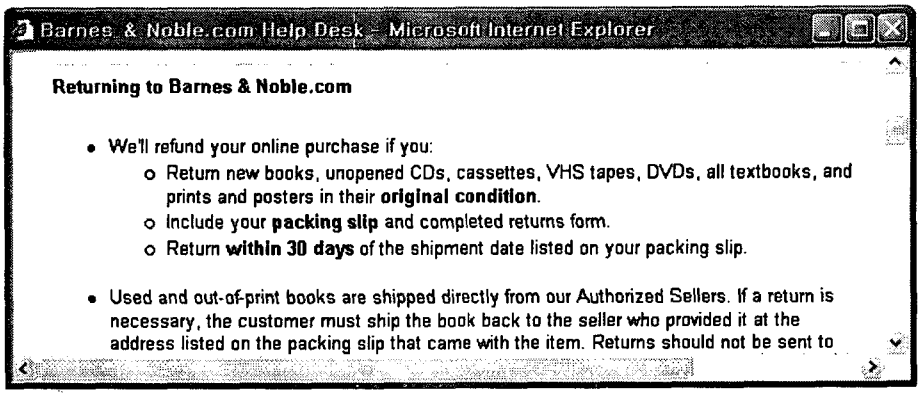

그림 3 – Barnes&Noble.com의 환불정책을 기술하고 있는 웹 페이지

2. 온톨로지 기반의 규칙 식별 개요

이 장은 온톨로지를 이용한 규칙습득의 기본적인 아이디어를 설명하고 있다. 2.1에서는 규칙구성요소의 습득에 대해 간략히 설명하고, 2.2에서는 식별된 규칙구성요소들로부터 규칙을 생성하는 방법에 대해 설명하고자 한다.

2.1. 온톨로지 기반의 규칙구성요소 식별

그림 3의 BarnesAndNoble.com (BN) 웹 페이지는 책에 대한 반송관련 정책을 설명하고 있다. 만일 지식관리자가 *books* 혹은 *CDs*가 변수값이라는 사실을 알고 있으면 그는 *<value>books</value>* 혹은 *<value>CDs</value>* 과 같은 태그를 이용하여 각각 변수값이라는 사실을 표현해 놓을 수 있다. 이것이 규칙식별의 간단한 예이다.

```
{{ "days of the shipment"
   IS-A: Variable
   Values: 30  }}
{{ "Returned status"
   IS-A: Variable
   Values: Opened "Obvious sign of use" "Original condition" }}
{{ Item
   IS-A: Variable
   Values: Book CD DVD "VHS tape" }}
{{ Refund
   IS-A: Variable
  V alues: Full Partial }}
```

그림 4 – 변수와 변수값 온톨로지의 예

그렇다면 규칙구성요소의 식별과정에서 온톨로지는 어떻게 활용될 수 있는가? 만일 우리가 그림 4와 같은 변수와 변수값에 관련된 온톨로지를 갖고 있다면 이를 이용하여 그림 3으로부터 규칙구성요소를 식별하는 것이 가능하다. 온톨로지의 표현은 RDF나 OWL로 쉽게 변환될 수 있으며 파악하기 쉬운 프레임 구조로 표현되었다. 예를 들어, 그림 4의 온톨로지로부터 우리는 그림 3에서 *refund*와 *days of the shipment*가 변수이고 *b ooks, CDs, VHS tapes* 등이 변수값임을 쉽게 파악할 수 있다. 그림 4의 온톨로지 예제는 간단해 보이나, 실제로 온톨로지를 통해 더욱 많은 정보를 제공하는 것이 가능하다. 예를 들어, 온톨로지를 이용하면 생략된 변수를 찾아내는 것도 가능하다. 그림 4에서 *item*은 *books, CDs, VHS tapes* 등의 변수값과 연결되는 변수이다. 그림 3에는 직접 *item*이 나오지 않지만 *books, CDs, VHS tapes* 등이 있기 때문에 *item*이 생략되어 있을 것이라고 어렵지 않게 짐작할 수 있다. 온톨로지를 이용한 규칙구성요소 식별의 결과물은 상호연결된 변수와 변수값 쌍의 집합이다. 이 변수와 변수값 쌍을 조합함으로써 *IF*와 *THEN*으로 구성된 규칙을 조합하는 것이 가능하다. 온톨로지 기반의 규칙구성요소 식별을 구현하기 위하여, 먼저 프레임 기반의 온톨로지를 OWL로 변환하고 Jena API를 이용해 파싱하였다. 그리고 스테밍 알고리즘(stemming algorithm)을 온톨로지와 웹 페이지 양쪽에 적용하여 비교할 단어들을 표준화하였으며 [12], 유의어 및 동의어를 검색하기 WordNet[16]을 이용해 단어를 확장하였다. 마지막으로 변수 및 변수값의 검색수준을 상위어 혹은 하위어까지 확장하기 위해 직접 고안한 시맨틱 유사도 척도를 이용하여 비교하였다. 본 논문에서는 초점을 흐리지 않기 위해 자세한 알고리즘[5]은 생략하였다.

2.2. 최적우선검색에 기반한 규칙 자동 생성의 개요

온톨로지를 활용한 자동화된 규칙 구성의 기본 아이디어는 동일한 도메인의 유사한 시스템에서 사용된 규칙들의 패턴을 이용하는 것이다. 즉 유사한 시스템에서 사용된 규칙을 일반화하여 온톨로지를 구축한다. 이 때 온톨로지 상의 규칙과 유사 시스템 상의 규칙의 관계는 1:1이 아닌 1:N의 관계가 성립되는데 이는 시스템에 사용된 여러 규칙이 온톨로지로 변환되는 과정에서 일반화되어 하나의 규칙으로 통합되기 때문 이다.

그림 5는 온톨로지 규칙의 예를 보여주고 있다. 만일 텍스트에서 식별된 변수들로부터 이 규칙과 유사한 패턴을 발견하면 최적우선탐색[18]을 활용하여 이 변수들로 구성된 하나의 규칙으로 생성하는 것이 가능하다. 예를 들어 설명하면, 웹 페이지로부터 *item, refund, days of the shipment* 변수를 식별한 경우 그림 5에 있는 규칙 온톨로지로부터 이 변수들이 *Return Policy Rule*로 구성될 수 있음을 알 수 있다.
최적우선탐색 알고리즘은 변수 인스턴스들로 이루어진 탐색트리를 생성한 후에, 트리에 있는 변수 인스턴스들을 온톨로지의 규칙들에 하나씩 할당해 나가면서 규칙을 생성한다. 또한, 트리로부터 가장 적절한 노드를 선택하기 위한 평가함수가 요구된다. 보다 상세한 알고리즘은 4장에서 기술하도록 하겠다.

{{ "Refund Policy Rule"
IS-A: Rule
IF: "days of the shipment" Item "Returned status"
THEN: Refund }}

그림 5 – 온톨로지 상의 규칙의 예

2.3. 규칙 온톨로지의 설계

본 논문에서는 규칙구성요소와 규칙의 구조에 대한 정보를 제공하는 규칙 온톨로지를 ***OntoRule***이라는 이름으로 설계하여 사용하였다. 규칙 구성요소 식별을 위한 온톨로지는 규칙, 규칙값, 그리고 이 둘간의 관계로 구성되나, 규칙 구성을 위한 온톨로지는 그림 5에서와 같이 일반화된 규칙 구조가 요구된다. 따라서 그림 5의 규칙은 *IF* 부분과 *THEN* 부분에 해당하는 변수를 포함하고 있다. 이와 같이 규칙과 변수와의 관계는 *Rule* 클래스의 *IF* 와 *THEN* 슬롯으로 표현된다. *AND*와 *OR* 같은 접속어(connective)는 복잡한 중첩구조를 가지고 있으며, 온톨로지에 모든 접속어를 표현하게 되는 경우 규칙에 대한 일반화 효과가 없어지기 때문에, 접속어의 표현은 OntoRule에 포함시키지 않았다.
다른 관점에서 보면 본 논문에서 제안하는 온톨로지 대신 직접 규칙베이스들의 규칙들을 규칙의 습득과정에서 이용하는 것이 가능할 수 있다. 그러나 규칙 습득에 필요한 정보들을 일반화하고 축약한 OntoRule을 규칙 습득에 이용하는 것에 비해, 규칙베이스의 규칙들을 직접 이용하는 것은 훨씬 더 많은 기억장소와 추가적인 처리과정이 요구될 것이다. 이러한 이유로 본 연구에서는 규칙 식별시 규칙베이스을 직접 이용하지 않고 이를 일반화한 OntoRule을 활용하였다.

3. 최적우선탐색 기반 규칙 자동 생성과 관련된 논점

본 장에서는 최적우선탐색을 활용하여 식별된 변수들로부터 규칙을 생성하는 방법에 대해 설명하고 이 과정에서 발생하는 다양한 이슈들에 대해 다루도록 하겠다.

3.1. 최적우선탐색을 통한 규칙 생성 방법

본 연구의 방법론을 소개하기에 앞서 변수(variable)와 변수 인스턴스(variable instance)의 차이를 구분하면, 변수 인스턴스란 주어진 웹페이지로부터 추출된 변수들을 의미하고 변수란 OntoRule 상의 규칙을 구성하고 있는 변수를 의미한다. 따라서, OntoRule 상의 하나의 변수에 대해 여러 개의 변수 인스턴스들이 웹 페이지에 존재할 수 있다.
본 연구에서 규칙을 생성하는 기본 아이디어는 웹 페이지로부터 식별된 변수 인스턴스들을 적절히 조합하여 규칙을 생성하는 것이다. 이를 위해 먼저 변수 인스턴스들과 온톨로지의 규칙들을 비교하여 적절한 규칙 후보들을 추출한다. 규칙 후보를 이루고 있는 각각의 변수들에 대하여 적절한 변수 인스턴스를 하나씩 할당함으로써 규칙을 완성해나갈 수 있다.
최적우선탐색 기법은 현재의 변수 인스턴스들을 탐색트리로 표현하고, 이 중에서 적절한 변수 인스턴스를 선택하는 과정을, 최적우선탐색의 평가함수를 이용함으로써 가능하게 한다. 즉, 현재의 변수 인스턴스가 규칙후보에 적절한지를 평가함수를 통해 평가함으로써 가장 적절한 변수 인스턴스들을 선택하여 규칙후보에 할당할 수 있다.
예를 들어, 변수 *A, B, C*로 이루어진 규칙후보 R_1 = {*A, B, C*}이 있고 변수 인스턴스들의 집합 VI = {*A1, A2, B1, C1, C2, C3*}이 있다고 가정하자. 여기서 *A1*과 *A2*는 변수 *A*의 인스턴스들이므로 규칙을 만들기 위해서는 둘 중 하나를 선택해야 한다. 또한 {*C1, C2 C3*} 중에서도 하나를 선택해야 한다. 이를 위해서는 어느 인스턴스가 보다 적절한지를 판단할 수 있는 평가함수가 요구되고 최적우선탐색은 이 평가함수를 이용하여 필요한 인스턴스들을 적절하게 규칙후보들에 배정할 수 있다.

3.2. 최적우선탐색과 관련된 논점

최적우선탐색을 이용하여 규칙을 생성하기 위해서는 몇 가지 해결되어야 할 논점(issue)들이 존재한다. 첫째, 규칙후보들에 대해 변수 인스턴스를 할당하는 데에는 수 많은 조합이 존재할 수 있다. 따라서 이로 인한 복잡도를 제어할 필요가 있다. 이를 위해 본 연구에서는 Constrained Heuristic Search [2] 연구로부터 ***변수순 위화(variable ordering)***라는 개념을 채용하였다. 변수 순위화는 다음에 할당할 변수를 선택하기 위한 방법이다. 현재 가장 널리 사용되는

변수 순위화의 방법은 배정할 변수값의 수가 가장 적은 변수를 우선적으로 선택하는 것이다. 본 연구에서도 같은 방법을 이용하였다. 즉, 변수를 각각의 규칙후보에 배정하고자 할 때, 현재 변수의 인스턴수 수가 가장 적은 변수로부터 차례로 적절한 인스턴스를 선택한다. 이렇게 함으로써 초기에 선택 가능한 대안의 수를 줄이는 것이 가능하다. 그리고 일단 변수 인스턴스가 규칙 후보에 배정되면 뒤로 가면서 탐색공간은 줄어들게 되므로 보다 수월하게 탐색을 진행하는 것이 가능하다. 이와 같은 방식으로 배정할 변수의 순서를 정하는 것을 본 논문에서는 변수 순위화라고 부른다.

각 규칙 내의 변수 순위화에 앞서 규칙 후보들의 순서를 결정하는 것 또한 중요하다. 각 규칙 후보들에 대하여 변수 인스턴스들을 다양하게 조합하는 것이 가능하다. 3.1절의 마지막 문단 예에서 보면 R_1에 대한 변수 인스턴스의 조합으로는 {*A1, B1, C1*}, {*A2, B1, C1*}, {*A2, B1, C2*} 등이 가능하다. 이와 같이 규칙 후보들 각각에 대해 가능한 조합의 수를 계산하여 그 숫자가 가장 적은 규칙후보부터 인스턴스의 배정을 시작한다. 이 과정을 ***규칙 순위화(Rule Ordering)***라고 부른다.

둘째 논점은 최적우선탐색의 평가함수를 정의하기 위한 적절한 방법이다. 어떻게 변수 인스턴스가 규칙 후보에 적합하다는 것을 평가할 것인가? 본 연구에서는 하나의 규칙을 구성하는 변수 인스턴스들이 웹 페이지에서 서로 주변에 모여 있을 것이라는 가정으로부터 평가함수를 정의한다. 즉 한 변수 인스턴스 *A1*이 다른 변수 인스턴스 *A2*에 비해 이미 규칙후보에 배정된 다른 인스턴스들에 더욱 가깝다면 *A1*이 더 적합하다고 판단한다. 이와 같은 가정은 본 논문에서 제안되는 방법론에서 매우 중요하게 사용된다.

3.3. 본 연구의 방법론을 위한 가정

여기서는 규칙 생성을 위한 최적우선탐색 알고리즘을 보다 단순하고 실현 가능하도록 하기 위해 몇 가지 가정을 하고자 한다. 첫째 가정은 하나의 변수 인스턴스는 오직 하나의 규칙에서만 사용된다는 것이다. 즉 동일한 변수 인스턴스가 두 개 이상의 규칙에서 공유되어 사용되지 않는다. 이는 이미 이전단계에서 공유되는 변수들에 대한 처리를 완료하였으므로 합당한 가정이 된다. 둘째 가정은 규칙을 구성하는 변수 인스턴스들이 웹 페이지 내에 모두 존재한다는 가정이다. 변수 인스턴스들이 생략되어 있으면 탐색 트리는 불완전한 노드들로 구성되게 된다. 이와 같은 경우에 대한 해결방안은 추후 연구에서 다루고자 한다. 셋째 가정은 온톨로지에는 포함관계에 있는 규칙은 존재하지 않는다는 것이다. 예를 들어, 온톨로지에 {*A, B, C*}라는 규칙이 존재하고 있으면 {*A, B*}는 존재하지 않는다. 이는 규칙의 일반화 과정에서 {*A, B*} 규칙이 {*A, B, C*}로 통합되기 때문에 합당한 가정이 된다. 이 가정으로 인해 동일한 변수 인스턴스 집합에 대해 {*A, B*}와 {*A, B, C*}를 동시에 추천하는 것을 방지할 수 있다.

4. 최적우선탐색을 이용한 규칙생성

이 장에서는 규칙온톨로지를 이용해 규칙을 생성하기 위하여, 추출된 규칙구성요소들을 조합하는 최적우선탐색 알고리즘에 대해 설명하고자 한다.

4.1. 규칙생성을 위한 준비단계

최적우선탐색 알고리즘의 입력은 웹 페이지에서 식별된 변수 인스턴스들이며 $VI=\{VI_1, VI_2, ..., VI_i, ..., VI_n\}$로 표기된다. 출력은 규칙들의 인스턴스들이며 $RI=\{RI_1, RI_2, ..., RI_p, ..., RI_q\}$로 표기된다. 규칙 RI_p는 규칙후보에 배정된 변수 인스턴스들의 집합으로 표현된다. 규칙생성과정에서 사용되는 온톨로지는 규칙후보들의 집합이며 $RC=\{R_1, R_2, ..., R_j, ...,R_l\}$로 표기되고, 각 규칙후보 R_j는 변수들의 집합으로서 $\{V_{j1}, V_{j2}, ..., V_{jk}, ..., V_{jm}\}$으로 표기된다. 즉, 규칙의 생성은 규칙후보 R_j에 있는 각 변수들에 대하여 적절한 변수 인스턴스들을 VI로부터 골라 배정하는 과정으로 이해될 수 있다.

본 논문에서 제안되는 방법론에 대한 이해를 돕기 위하여 다음과 같은 후보규칙 RC와 식별된 변수 인스턴스들 VI로 구성된 예제를 만들었다. 텍스트에서 변수 인스턴스들은 중복되어 사용될 수 있기 때문에 이를 표현하기 위하여 변수명 뒤에 번호를 붙임으로써 변수 인스턴스를 표기하였다. 예를 들어 변수 *B*의 첫째 인스턴스는 *B1*으로 표기되고 두번째 인스턴스는 *B2*로 표기된다. 다음의 VI는 R_2, R_3, R_1, R_4의 순서로 된 네 개의 규칙으로 구성되어 있다. 본 연구에서는 이들 네 개의 규칙이 제안된 알고리즘을 통해서 올바르게 생성되는지 확인할 것이다. 문제를 단순화하기 위하여 하나의 규칙을 이루는 모든 변수들이 텍스트에 생략되지 않고 서술되어 있으며, 하나의 변수가 하나의 규칙 내에서 반복되어 사용되지 않는 것으로 가정하였다.

$VI = \{B1, D1, A1, C1, D2, B2, C2, A2, E1, D3\}$
$RC = \{R_1, R_2, R_3, R_4\}$
$R_1 = \{A, B, C\}, R_2 = \{A, B, D\}, R_3 = \{C, D\}, R_4 = \{D, E\}$

이상의 예제로부터 우리는 다음과 같이 텍스트에 존재하는 해당 변수의 인스턴스들의 개수를 세는 함수인 $Count(V_{jk})$를 계산할 수 있다.

$Count(A) = 2, Count(B) = 2, Count(C) = 2, Count(D) = 3, Count(E) = 1$

4.2. 규칙 순위화 (Rule Ordering)

본격적인 탐색 알고리즘을 시작하기 위해, 우선 OntoRule로부터 규칙후보들을 추출하여야 한다.

다음 식은 그 과정을 설명하고 있다. 어떤 규칙 R_j의 변수들이 VI 내에 모두 존재하고 있다면 R_j는 규칙후보가 된다.

$$RC=\{R_1, R_2, ..., R_j, ..., R_l\} \mid R_j \subset VI\}$$

앞에 설명한 바와 같이 규칙 순위화는 규칙에 대해 만들어낼 수 있는 변수 인스턴스들의 조합이 작은 규칙부터 정렬하는 것이다. 규칙 순위화의 첫 단계는 다음과 같이 각 규칙들에 대하여 규칙생성을 위해 가능한 변수 인스턴스 조합의 수를 계산하는 것으로 다음 식과 같이 표현된다.

For each R_j in RC,

$$NC(R_j)=\prod_{k=1}^{m} Count(V_{jk})$$

예를 들어, 규칙 R_2에 대한 변수의 조합을 VI에 있는 변수 인스턴스들로부터 만들어내고자 한다면, $\{A1, B1, D1\}$, $\{A2, B1, D1\}$, $\{A1, B2, D2\}$들이 규칙 R_2에 대해 가능한 변수 인스턴스 조합의 예가 된다. 이 때 변수 A는 VI에 $A1$, $A2$ 두 개의 인스턴스를 갖고 있고, B는 두 개, D는 세 개의 인스턴스를 갖고 있으므로, 가능한 조합의 수를 나타내는 $NC(R_2)$은 2*2*3의 계산을 통해 12가 된다.
다음은 예제에 있는 모든 규칙들에 대해 $NC()$를 계산한 것이다.

$NC(R_1)$ = 2*2*2 = 8, $NC(R_2)$ = 2*2*3 = 12, $NC(R_3)$ = 2*3 = 6, $NC(R_4)$ = 3*1 = 3

규칙 순위화의 마지막 단계는 $NC()$의 증가 순으로 RC의 규칙후보들을 정렬하는 것이다. 예제에서 정렬된 순서 $RuleOrder(RC)$는 $\{R_4, R_3, R_1, R_2\}$이다.

4.3. 변수 순위화 (Variable Ordering)

규칙 순위화에 의해 규칙순서가 결정되면 다음 과정은 각 규칙 내에서 변수들의 순서를 결정하는 것이다. 이를 변수 순위화라 하고 규칙 순위화와 동일한 휴리스틱을 적용하였다. 즉, 변수 순위화에서는 가장 적은 변수 인스턴스를 갖고 있는 변수들부터 정렬하였다. 이를 위해 우선 각 변수에 대해 $Count(V_{jk})$를 계산하고, $Count()$의 증가 순으로 각 규칙 내에서 변수들을 정렬하였다. 본 예제에서는 $Count(D)$ = 3, $Count(E)$ = 1이므로 $VariableOrder(R_4)$ = $\{E, D\}$가 된다.
$RuleOrder$와 $VariableOrder$를 조합함으로써 전체의 탐색순위를 나타내는 $TotalOrder(RC)$를 생성하는 것이 가능하다. 예제에서 $TotalOrder(RC)$는 $\{\{E, D\}, \{C, D\}, \{A, B, C\}, \{A, B, D\}\}$이 된다. 최적우선탐색은 $TotalOrder(RC)$에 있는 규칙과 변수의 순서에 따라 진행된다.

4.4. 생성된 규칙에 대한 평가함수

3.2에 기술된 바와 같이, 규칙후보에 대한 변수 인스턴스의 적합도는 그 규칙후보를 구성하기 위해 이미 배정된 변수 인스턴스들과 새 인스턴스의 거리를 기반으로 계산된다. 이 거리를 쉽게 표현하기 위하여 분산을 사용하였다. 즉, 예를 들어 규칙후보 R_1에 대해 $C1$, $C2$ 중 어느 것이 더 적합한지 알아 보기 위해 먼저 $C1$의 적합도를 계산하고자 하는 경우, 만일 $\{A2, B1\}$가 이미 R_1에 배정되어 있다면, $C1$의 적합도는 $\{Pos(A2), Pos(B1), Pos(C1)\}$의 분산으로 표현될 수 있다. 여기서 $Pos(C1)$은 텍스트에서 $C1$의 위치를 의미한다. 예제에서 $C1$은 VI에서 넷째에 위치하고 있으므로 $Pos(C1)$은 4가 된다. 다음 $C2$의 적합도를 $C1$과 비교하고 싶다면 $\{Pos(A2), Pos(B1), Pos(C2)\}$의 분산을 구한 후에 앞서 구한 분산과 비교하면 된다. 분산이 작을수록 변수 인스턴스들이 가까이 모여 있다는 것을 의미하므로 적합도는 더욱 높게 된다.
규칙후보 R_j에 대한 $C1$의 적합도는 $Var(R_j)$로 표현되며 다음과 같이 정의된다.

$Var(R_j)$ = $Var(Pos(V_{j1}), Pos(V_{j2}), ..., Pos(V_{jk}), ..., Pos(V_{jm}))$, where $Pos(V_{jk})$ is a position of assigned instance of V_{jk} in the text.

그러나, RC에는 여러 규칙후보들이 존재하고 있으므로 $C1$의 적합도를 계산하기 위해서는 현재 $C1$이 포함되어 있는 규칙 외에도 $C1$에 도달하는 경로에 존재하는 모든 규칙들에 대해 그 분산을 계산하여야 한다. 예를 들어, 현재 C1까지의 경로가 $\{\{E1, D1\}, \{C2, D2\}, \{A2, B1, C1\}\}$인 경우, 앞서 계산된 $TotalOrder(RC)$=$\{\{E, D\}, \{C, D\}, \{A, B, C\}, \{A, B, D\}\}$를 기반으로 R_1에 대한 $C1$의 적합도를 계산하고자 한다면, $Var(R_1)$ 외에도 $\{E1, D1\}$과 $\{C2, D2\}$에 대한 분산을 추가로 계산 하여야 한다.
따라서 현재까지 구성된 경로에 대한 $C1$의 적합도는 다음과 같은 평가함수로 계산될 수 있으며 A* 알고리즘의 경우에는 현재까지 소요된 경비 $g(n)$으로 해석될 수 있다.

$$f(n) = Var(Path(n)) = \sum_{j=1}^{l} Var(R_j)$$

4.5. 최적우선 탐색 알고리즘

그림 6은 4.4절에서 설명된 평가함수를 사용하여 규칙을 생성하는 최적우선 탐색 알고리즘을 상세히 보여주고 있다. 그림 6에서 1번 라인인 *Initialization* 과정에서는 4.2절과 4.3절에서 설명된 바와 같이 먼저 온톨로지로부터 규칙후보들을 추출하고, 규칙 순위화와 각 규칙에 대한 변수 순위화를 실시하여 $TotalOrder$를 생성한다. 그림 6의 5번 라인에서 $firstVIs$는 최초의 ***OPEN*** 집합을 생성하기 위해 $TotalOrder$에 따라 VI로부터 선택된 변수 인스턴스이다.

```
1  Initialization:
2  choose candidate rules;
3  rule ordering;
4  variable ordering within each rule;
5  get firstVIs such that Variable(VI) = firstVar(TotalOrder) for each VI ∈ firstVIs;
6  OPEN = firstVIs; CLOSED = {};
7  begin
8  repeat
9      choose a currentVI from OPEN having the lowest f value;
10     if nextVar(currentVI, TotalOrder) is empty, then finished = true;result=currentVI;
11     else
12     begin
13       transfer currentVI from OPEN to CLOSED;
14       construct nextVIs such that
15         For each VI ∈ nextVIs,
16           Variable(VI) = nextVar(currentVI, TotalOrder) and VI ∉ path(currentVI);
17       set OPEN = OPEN ∪ nextVIs;
18   end;
19 until finished;
20 output path(result);
21 end;
```

그림 6 - 규칙생성을 위한 최적우선탐색 알고리즘

즉, *TotalOrder*의 첫째 변수와 매칭되는 변수 인스턴스를 *VI*로부터 추출하여 ***OPEN*** 집합에 넣게 된다. 예제에서는 *E1*이 유일하게 *TotalOrder*의 첫 변수인 *E*의 인스턴스이므로 {*E1*}이 *firstVIs*가 되고 다음 6번째 라인에서 ***OPEN*** 집합에 들어가게 된다. 그림 6에서 ***Repeat*** 구문의 첫 단계인 9번 라인은 평가 함수 *f(n)*에 따라 ***OPEN*** 집합으로부터 가장 적합한 변수 인스턴스를 골라내는 과정이다. *TotalOrder(RC)*에 있는 모든 변수들에 대해 변수 인스턴스를 배정하게 되면, 즉 더 이상 배정해야 할 변수가 없게 되면 알고리즘은 끝나고 현재까지의 경로를 규칙생성에 대한 결과로 출력하게 된다. 예제에서는 ***Repeat*** 구문의 첫 주기에서 *E1*이 선택된다. 선택된 노드는 그림 6의 13번 라인에서와 같이 ***CLOSED***로 옮겨지고, 14번 라인에서는 다름 주기에서 탐색 대상이 되는 노드들을 *VI*로부터 선택한다. *nextVar(currentVI, totalOrder)*는 *totalOrder*에서 *currentVI* 다음에 있는 변수를 의미한다. 그림 7은 탐색에서 사용되는 탐색트리의 일부와 탐색순서를 보여주고 있다. *E1*로부터 선택될 수 있는 다음 노드들은 *TotalOrder*에 따라 {*D1, D2, D3*}이다. 따라서 이들이 그림 6의 17번 라인에서와 같이 ***OPEN*** 집합에 포함되고 다음 ***Repeat*** 순환이 시작된다. 여기서 다음 노드들을 선택할 때 주의할 점은 현재 경로에 이미 존재하는 변수 인스턴스는 대상이 되지 않는다는 것이다. 경로에 존재한다는 것은 이 인스턴스가 이미 다른 규칙에 배정되었다는 것을 의미하고 따라서 다시 대상이 될 수는 없다. *path(currentVI)*는 *currentVI*의 현재까지의 경로를 반환한다. 예를 들어, 그림 7에서 깊이 3의 *C1* 밑에는 *D3*가 포함되지 않는다. *D3*가 이미 *C1*의 상위에 존재하고 있기 때문이다.

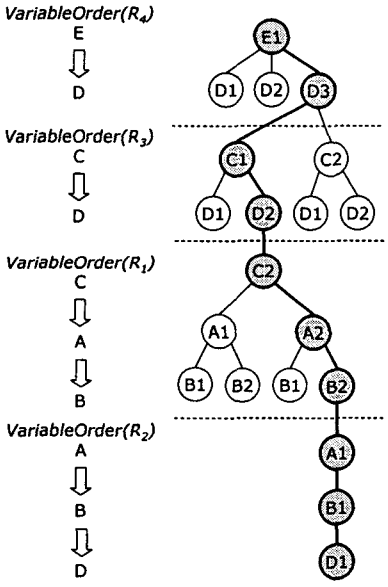

그림 7 - 예제에 대한 최적우선탐색의 탐색트리와 결과

이 알고리즘의 중요한 특성 중 하나는 ***OPEN*** 집합에 동일한 변수 인스턴스들이 들어있을 수 있다는 점이다. 예를 들어 그림 7에서 깊이 2의 *D1*과 깊이 4의 *D1*은 ***OPEN*** 집합에 서로 다른 대안으로서 함께 있을 수 있다. 깊이 2의 *D1*은 R_4에 배정되고 깊이 4의 *D1*은 R_3에 배정되기 때문이다. 일반적인 최적우선탐색에서는 더 우월한 값을 가진 노드가 ***OPEN*** 집합에 있는 동일한 다른 노드에 겹쳐서 쓰여지게 된다. 그러나, 본 알고리즘은 두 노드가 서로 구분되어야 하므로, 이 알고리즘에서는 각 노드의 키로 그 노드의 경로를 사용하였다. 예를 들어, 깊이 2의 *D1*의 키는 {{*E1, D1*}}이 되고 깊이

4의 왼쪽 *D1*의 키는 {{*E1, D3*}, {*C1, D1*}}이 된다.
알고리즘이 *TotalOrder(RC)*에 있는 모든 변수에 인스턴스를 배정하고 나면, *currentVI*의 경로를 출력하고 알고리즘을 종료한다. 예제에서는 *D1*이 마지막 *currentVI*가 되고 출력된 경로는 {{*E1, D3*}, {*C1, D2*}, {*A2, B2, C2*}, {*A1, B1, D1*}}이며 이 경로는 후보 규칙 {R_4, R_3, R_1, R_2}에 대한 최종생성 규칙이 된다.

4.6. 규칙을 구성하는 변수 및 변수값에 대한 *IF*와 *THEN*의 결정

최적우선탐색에 의해 규칙후보들에 대한 변수 인스턴스들의 배정이 완결되면, 각 변수 인스턴스들이 *IF*에 속해야 하는지, *THEN*에 속해야 하는지 결정해야 한다. 이 과정은 매우 단순하다. 만일 변수 인스턴스와 매칭되는 변수가 온톨로지의 규칙후보에서 *IF*에 속한다면 *IF* 를 배정하고, *THEN*에 속한다면 *THEN*을 배정한다. 변수값에 대해서도 동일하게 *IF*와 *THEN*을 배정한다.

4.7. 최적우선탐색과 관련된 추가 논점

4.5에서 설명된 최적우선탐색 알고리즘에서 고려되지 않은 몇 가지 논점들이 있다. 우선 알고리즘에서는 모든 변수들이 *VI*에 매칭되는 규칙들만을 다음 공식을 통해 규칙후보로 선정하였다.

$$RC=\{R_1, R_2, \ldots, R_i, \ldots, R_k \mid Var(R_i) \subset VI\}$$

그러나 사이트마다 규칙이 변경될 수 있으므로 이전 사이트에서 사용되었던 어떤 규칙이 현재 사이트에서는 부분적으로만 사용될 수 있다. 예를 들어 *I F A and B THEN C* 형태의 규칙이 {*A, B, C*}로 온톨로지에 포함되어 있고, *VI*는 {*A1, C1, D1, E1*}이라면, 현재 알고리즘에서는 이 규칙이 규칙후보가 되지 않는다. 그러나, 실제로는 *IF A THEN C* 의 형태로 현재 사이트에서 사용되고 있을 수 있다. 이와 같은 경우를 고려하기 위하여 다음과 같이 현재의 알고리즘을 확장할 수 있다.

$$RC=\{R_1, R_2, \ldots, R_i, \ldots, R_k \mid Var(R_i) \cup VI \neq \Phi\}$$

즉, 규칙의 일부가 *VI*에 포함되어 있으면 규칙후보로 추천된다.
또한 규칙 순위화에서 $NC(R_j)$을 계산하는 부분도 다음과 같이 수정되어야 한다. 즉 $Count(V_{jk})$가 0인 변수에 대해서는 $NC(R_j)$의 계산에 포함하지 않는다.

$$NC(R_j) = \prod_{k=1, Count(V_{jk})\neq 0}^{m} Count(V_{jk})$$

여기에 추가하여, 변수 순위화에서는 $Count(V_{jk})$가 0인 변수를 순위에서 아예 제외한다. 그 결과, 존재하지 않는 변수 인스턴스에 대해서는 규칙후보에 배정할 필요가 없어 진다.
이상과 같이 알고리즘을 변경할 경우, 매우 중요한 부분은 규칙후보가 될 수 있는 규칙들에 대해 적절한 기준을 설정하는 것이다. 예를 들어 이상의 식으로는 단지 하나의 변수만 매칭되어도 규칙후보가 될 수 있다. 따라서 기본적으로 본 연구에서는 규칙후보가 되기 위한 조건으로 적어도 *IF*와 *THEN*절에 각각 하나 이상의 매칭 변수가 있어야 한다는 것을 정의하였다.
현재의 알고리즘이 완전하지 않기 때문에 이 외에도 다양한 논점이 있을 수 있으며, 향후 연구에서 이와 같은 부분들을 보다 깊이 있게 다루고자 한다.

5. 온톨로지를 이용한 규칙 식별의 성능 평가

본 절에서는 OntoRule 을 이용한 자동화된 규칙 식별의 효과를 측정함으로써 4 장에서 제안된 본 연구의 방법론의 성능을 평가하고자 한다. 본 실험을 위해 규칙 식별에 이용할 온톨로지가 필요하므로, 먼저 Amazon.com 으로부터 온톨로지를 구축하였다. 이렇게 구축된 온톨로지는 널리 알려진 온라인 서점인 Barnes&Noble.com (이하 BN)과 Powells.com (이하 Powells)로부터 규칙을 습득하는데 적용되었다.
본 연구의 성능을 평가하기 위해 두개의 측정 변수를 정의하였다. 먼저 첫번째 측정 변수는 정확율(precision ration, ***PR***로 표시)로서 온톨로지를 통해 정확하게 추천된 갯수를 온톨로지에 의해 추천된 갯수로 나눈 값이다. 따라서 ***PR***의 값이 1.0이라는 것은 온톨로지에 의해 추천된 것들이 모두 정확하다는 것을 의미한다. 두번째 측정변수는 재현율(recall ration, ***RR***로 표시)로서 온톨로지를 통해 정확하게 추천된 개수를 실제로 식별되어야 할 실제 개수로 나눈 값이다. ***PR***과 ***RR*** 값 모두 높게 나올수록 추천의 정확도가 높다는 것을 의미한다.
규칙 생성과 *IF/THEN* 식별 실험에서 표 1과 같이 매우 만족할 만한 실험결과를 얻을 수 있었다. 표 1을 살펴보면 BN에서 규칙의 ***RR*** 값은 다른 숫자에 비해 매우 낮게 나타났는데 그 이유는 다음과 같다. 현재 알고리즘에서는 한 행에 여러 규칙이 나타나는 경우를 처리할 수 없다는 한계점이 존재하는데, BN의 경우는 테이블의 각 행에 여러 개의 규칙을 표현하는 방식을 취하고 있어서 그러한 규칙들을 식별할 수 없었기 때문이다. 또한 표1에서 BN의 *IF*의 ***PR***과 ***RR***의 값이 상대적으로 낮게 나타났는데, 그 이유는 BN은 배송비 표에서 배송 항목에 대한 중복된 설명을 넣어 이전 규칙에 비해 *IF* 부분에 새로운 변수들이 추가되었는데 이는 기존의 규칙 온톨로지에 없는 내용이므로 추천될 수 없었기 때문이다.

표 1 - 규칙 생성과 IF/THEN 식별 결과

	Rule		IF		THEN	
	PR	RR	PR	RR	PR	RR
BN	100.0%	70.6%	59.4%	41.8%	100.0%	96.6%
Powells	100.0%	99.6%	100.0%	98.2%	100.0%	99.1%

본 실험의 주요 한계점은 실험에 사용된 예제 사이트가 두개밖에 없다는 점이다. 이에 본 실험의 목적은 우리가 제안한 방법론에 대한 유효성을 실험적으로 정확히 입증하고자 하는 것이 아니라 예를 통해 본 방법론이 잘 동작하고 있음을 보여주는 것에 있다. 그러므로 본 연구의 방법론을 입증하고 일반화할 수 있도록 좀 더 충분한 웹사이트를 확보해 실험하는 것이 요구되며 현재 추후 연구로 서 계획 중에 있다.

5. 결론

규칙 습득을 위한 이전의 XRML 접근방법은 식별된 변수와 변수값으로부터 규칙을 생성하기 위해 지식 관리자의 수작업이 많이 요구되었다. 따라서, 본 연구에서는 온톨로 지를 활용하여 최적 우선 탐색을 통해 자동으로 규칙을 생성하는 방법론을 제안하였다.

최적 우선 탐색 알고리즘을 설계하는 과정에서 탐 색의 복잡도를 줄이기 위해 변수 순위화(variable ordering)와 규칙 순위화(rule ordering) 기법을 채용하였다. 변수 순위화란 한 규칙 내에서 변수의 탐색 순서를 결정하는 과정으로 탐색 순서는 가장 적은 변수 인스턴스를 가지고 있는 변수로부터 시작되는데 이는 초기에 선택가능한 대안의 숫자를 줄여줄 수 있다. 이와 유사하게 규칙 순위화란 규칙의 탐색 순서를 결정하는 과정으로 규칙을 만들 수 있는 변수 인스턴스들의 가능한 조합의 숫자가 가장 적은 규칙을 먼저 선택하도록 하였다. 이 외에도 규칙후보에 대한 변수 인스턴스의 적합도를 평가하기 위해 평가함수를 고안하였다. 본 연구에서 제안된 평가함수는 한 규칙을 구성하고 있는 변수들은 텍스트 상에서 서로 주변에 모여 있을 것이라고 가정을 하고 그 규칙후보를 구성하기 위해 이미 배정된 변수 인스턴스들과 새 인스턴 스 사이의 거리를 계산한다.

본 연구에서 제안된 방법론의 한계점 및 앞으로 수행되어야 할 연구는 다음과 같다. 첫째 본 연구에서 제안된 알고리즘의 성능을 평가하기 위해 체계적인 실험이 도입되어야 한다. 둘째, 본 연구에서 제안된 방법론은 기초적 인 것으로서 실제 응 용에 적용할 수 있도록 본 연구의 방법론을 좀 더 일반화해야 한다. 이를 통해 본 연구의 방법론이 웹으로부터 규칙을 습득하는 효율적 이고 새로운 방법이 될 것으로 기대한다.

Acknowledgement

본 연구는 21세기 프론티어 연구개발 사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스 컴퓨팅 및 네트워크원천기반기술개발사업의 지원에 의한 것임.

References

[1] Alani, H., Kim, S., Millard, D.E., Weal, M.J., Hall, W., Lewis, P.H., and Shadbolt, N.R.,: Automatic Ontology-Based Knowledge Extraction from Web Documents. IEEE Intelligent Systems 18(1), (2003) 14-21.

[2] Beck J. C., and Fox. M.: A Generic Framework for Constraint Directed Search and Scheduling. AI Magazine, 19(4) (1998) 101-130.

[3] Berners-Lee, T., Hendler, J., and Lassila, O.: The Semantic Web. Scientific American (2001).

[4] Brickley, D., and Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, < http://www.w3c.org/TR/rdf-schema/> (2004).

[5] Chae S., Kim W., and Park S.: Extracting Rule Components from the Web using Ontology. Working Paper (2006).

[6] Crow, L., and Shadbolt, N.: Extracting Focused Knowledge from the Semantic Web. International Journal of Human-Computer Studies 54, (2001) 155-184.

[7] Decker, S., Erdmann, M., Fensel, D., and Studer, R.: Ontobroker: Ontology based Access to Distributed and Semi-Structured Information. In R. Meersman et al. (eds.), Database Semantics, Semantic Issues in Multimedia Systems, Kluwer Academic Publisher, Boston (1999).

[8] Donini, F.M., Lenzerini, M., Nardi, D., and Schaerf, A.: Reasoning in Description Logics, Principles of Knowledge Representative. Ed: G. Brweka,. CSLI Publications, Stanford, (1996) 191-236.

[9] Grosof, B.N., Horrocks, I., Volz, R., and Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logic, In Proceedings of 12th International Conference on the World Wide Web (WWW-2003), Budapest, Hungary (2003).

[10] Hemnani, A., and Bressan, S.: Extracting Information from Semi-Structured Web Documents. Lecture Notes in Computer Science 2426 (2002) 166-175.

[11] Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., and Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, <http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/> (2004).

[12] Jones, K.S., and Willet, P. (eds.): Readings in Information Retrieval, Morgan Kaufmann Publishers, San Francisco (1997).

[13] Kang, J., and Lee, J.K.: Rule Identification from Web Pages by the XRML Approach. Decision Support Systems 41(1), (2005) 205-227.

[14] Lee, J.K., and Sohn, M.: Extensible Rule Markup Language – toward Intelligent Web Platform. Communications of the ACM 46, (2003) 59-64.
[15] Manola, F., and Miller, E.: Resource Description Framework (RDF) Primer. W3C Recommendation, <http://www.w3.org/TR/REC-rdf-syntax/> (2004).
[16] Miller, G.A.: WordNet a Lexical Database for English, Communications of the ACM 38(11) (1995) 39-41.
[17] Park, S., Lee., J.K., Kang, J.Y.: The Effect of Knowledge Acquisition through OntoRule: XRML Approach. Journal of Korea Intelligent Information Systems Society 11(2) 151-173.
[18] Pearl. J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, Reading, MA (1984).
[19] Reynolds, D.: Jena 2 Inference Support. <http://jena.sourceforge.net/inference > (2005).
[20] RuleML: The Rule Markup Initiative. <http://www.dfki.uni-kl.de/ruleml/> (2003).
[21] Schmidt, G., and Wetter, T.: Using Natural Language Sources in Model-Based Knowledge Acquisition. Data & Knowledge Engineering 26 (1998) 327-356.
[22] Smith, M.K., Welty, C., and McGuinness, D.: OWL Web Ontology Language Guide. W3C Recommendation, < http://www.w3c.org/TR/owl-guide/> (2004).
[23] Volz, R., Oberle, D., Staab, S., and Motik, B.: KAON SERVER - A Semantic Web Management System. In Alternate Track Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, 20-24 May 2003. ACM (2003).
[24] Yang, J., Oh, H., Doh, K.G., and Choi, J.: A Knowledge-Based Information Extraction System for Semi-structured Labeled Documents. Proceedings of the 4th Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science 2412 (2002) 105-110.
[25] Zou, Y., Finin, T., and Chen, H.: F-OWL: an Inference Engine for Semantic Web. Proc. 3rd NASA/IEEE Workshop on Formal Approaches to Agent Based Systems, FAABS III, 16-18 April 2004, Greenbelt MD, Lecture Notes in Computer Science, v. 3228, Springer Verlag (2004).