어휘 의미 패턴(Lexico-Semantic Pattern)과 온톨로지를 이용한 정보검색기의 설계 및 구현

The Design and Implementation of an Information Retrieval System Using Lexico-Semantic Pattern and Ontology

김병우 Byoungwoo Kim*, 고영중 Youngjoong Ko*
*동아대학교 컴퓨터공학과

요약 본 논문에서 제안하는 정보 검색기는 일반적인 불리언(Boolean) 질의를 통해서 정보를 검색하는 것이 아니라, 문장으로 입력된 질의형태의 패턴을 분석하여 그에 맞는 정보를 직접 제공하는 것에 목적을 둔다. 이를 위해 어휘 의미 패턴(Lexical Semantic Pattern)과 온톨로지(Ontology) 기술이 정보검색기 개발에 적용되었다. 제안된 시스템에서는 다양한 형태로 표현된 문장 질의를 어휘 의미 패턴을 사용해서 문장의 질의 패턴을 추출하고 사용자 질의를 하나의 온톨로지(Ontology) 추론 질의와 매칭함으로써 질의에 대한 정확한 해답을 추출할 수 있다. 또한, 자연어 문장 입력에 대한 검색 질의 생성기를 구축하고 온톨로지로 표현된 지식을 사용하여 정보검색기 질의를 자동으로 확장함으로써 더욱 정확한 정보 검색 결과를 만들어 낼 수 있다.

핵심어: 정보검색기(Information Retrieval System), 어휘의미패턴(Lexico-Semantic Pattern), 온톨로지(Ontology)

1. 서론

21세기에 들어와서 시멘틱 웹이 대두되어지면서, 지식을 어떻게 표현 할 것인가에 대한 관심이 높아 지고 있다. 데이터베이스를 사용하여 지식에 대한 정보를 표현하였던 이전과 달리 요즘은 온톨로지를 이용하여 정보를 표현하는 시도가 많아지고 있다[1]. 이러한 온톨로지로 표현된 지식은 데이터베이스에서 표현된 지식보다 적은 정보를 이용하여 더 많은 정보를 온톨로지 추론 과정으로 얻을 수 있다는 장점을 가지고 있다.

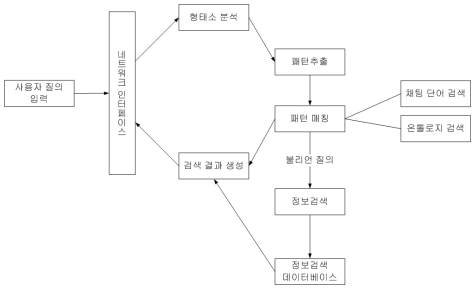
제안된 정보검색기는 사용자가 얻고자 하는 정보에 대한 질의를 하면 그에 맞는 정확한 응답을 제공하는 기능이 제공된다. 이러한 정보검색기를 사용한다면 사용자가 단어 형태로 정보를 검색하는 것이 아니라 필요한 정보에 대해 자연어로써 직접 사람에게 물어보는 것처럼 질문을 시스템에게 할 수 있게 된다. 문장형태의 사용자 질의는 같은 질의를 여러 가지 형태로 표현 가능하지만 같은 질의일 경우 비슷한 의미 패턴으로 구성되는데 이러한 패턴을 어휘 의미 패턴(Lexico-Semantic Pattern(LSP))[2]이라 부른다. 이러한 의미 패턴을 사용함으로써 다양한 표현형태를 보다 작은 소그룹의 질문으로 정규화 시킬 수가 있게 된다.

정규화된 사용자 질의는 데이터베이스 쿼리로 만들 수가 있게 되는데[3] 여러 개의 사용자 질의는 하나의 데이터베이스 쿼리로 대응되게 된다. 이렇듯 여러가지 질의의 수를 줄임으로써 데이터 처리의 양을 줄일 수 가 있게 되는 것이다.

또한 정규화된 정보검색질의는 자연어로 입력되어 있기 때문에 질의의 모호성이 존재 한다. 이런 질의의 모호성을 온톨로지를 이용하여 질의를 자동으로 확장하여 더욱 정확한 정보검색 결과를 제공 하게 되는 것이다.

본 논문에서는 LSP을 이용하여 사용자 질의를 정규화시키고, 온톨로지로 표현된 지식을 검색하기 위해 질의를 생성 및 확장하여 사용자에게 제공할 수 있는 정보검색 모델을 제안한다.

2. 전체 시스템의 구성



[그림 1] 제안된 정보검색 시스템의 구성도

[그림 1]에서는 전체 시스템 구성도를 보인다. 사용자의 질의는 네트워크를 통해 검색 엔진으로 들어오게 된다. 들어온 사용자 질의를 LSP로 만들기 위해 형태소분석 후 의미 패턴을 추출 하게 되는데 추출된 의미단어는 LSP의 구성요

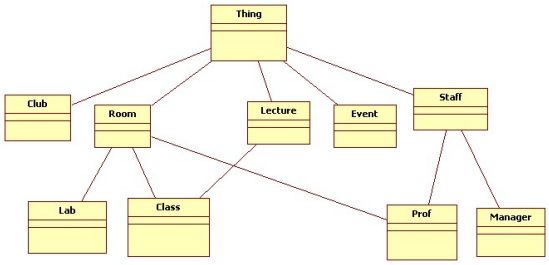
소가 된다. 이렇게 만들어진 LSP는 정규표현식(Regular Expression)으로 만들어 이미 만들어 놓은 예상 질문의 LSP 패턴과 비교 작업을 한다. 이러한 매칭은 두가지 모듈에서 이미 개발자가 정의해 놓은 우선순위에 의해 하나씩 검색 한다. 패턴 매칭 우선순위는 채팅문장 → 온톨로지 쿼리의 순으로 매칭 한다. 매칭되는 결과가 없을 경우에는 사용자 문장에서 LSP구성 단어들로 생성된 불리언 질의를 통해 정보검색엔진에서 나온 결과를 사용자에게 보여지게 된다.

정보검색엔진은 Lucene 2.0 [4]을 사용 하였다. 이 정보검색엔진을 이용하여 온톨로지를 검색하여 질의의 확장을 수행하여 질의를 생성하고 생성된 질의는 정보검색 결과를 더욱 높여 주는 역할을 할 수 있다.

3. 온톨로지

3.1 온톨로지 모델링

온톨로지를 기반하고 있는 정보검색기의 경우 데이터베이스와 달리 온톨로지가 얼마나 잘 구성 되어 있는가에 따라 시스템의 성능이 좌우 되는 특징을 가지고 있다. 그래서 온톨로지를 어떻게 구축할 것인가에 대한 전략이 가장 많은 비중을 차지 한다. 온톨로지의 특성상 표현된 지식에서 중첩된 질의문을 생성하여 여러가지 지식 추론이 가능하다. 즉, 두가지 이상의 관계(relation)로 여러가지 데이터의 추론이 가능하다. 온톨로지에 표현되어져 있는 내용은 예상되는 사용자의 질의를 기반으로 만들어져 있다.



[그림 2] 동아대 컴퓨터공학과 정보검색기에 사용된 온톨로지

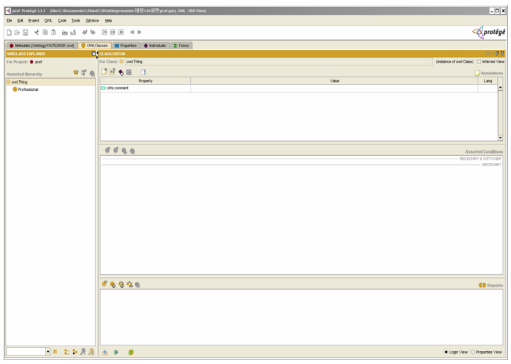
[그림 2]는 동아대학교 컴퓨터공학과에 대한 온톨로지를 간략히 표현한 것이다. 이와 같이 온톨로지를 구축하면 Racer[5,6] 프로그램을 이용하여 각 개념에 들어 있는 내용들이 추론 가능 하다.

3.2 온톨로지 구축

앞서 소개한 온톨로지 모델을 구현 하기 위해선 온톨로지 제작 툴이 필요하다. 직접 구현한 온톨로지는 Protégé 3.1[7]을 이용 하여 구축을 하였다. Protégé 3.1은 자바로 구현 되어 있어 플랫폼의 제약이 거의 없다. 현재 나와있는 버전은 Windows, Linux, HP-UX, Unix등에 사용 가능하도록 홈페이지에서 제공한다.

온톨로지는 개념(Cocept), 관계(Relation), 개체(Individual), 속성(Attribute)이 4가지로 구성이 된다. 개념은 어떠한 지식의 개념을 말하는 것이며, 관계는 각 개념간 연관성을 표현한다. 또한 속성은 그 클래스에 속해있는 속성값을 말하며, 개체는 클래스의 객체라고 보면 된다.

예를 들어 교수라는 클래스에서 속성은 이름, 전공, 등이 되는 것이고, 개체는 이러한 속성을 가진 한 명의 교수를 의미하게 되는 것이다.



[그림 3] Protégé 3.1 실행화면

아래는 [그림 3]에서 보여진 Protégé 3.1을 사용하여 Professional이라는 클래스를 간략하게 만든 OWL 파일이다. Professional은 Name과 Major를 속성으로 가지고 있고, Major, Name이 두 가지 속성은 String Type으로 타입이 설정되어져 있는 모습을 볼 수가 있다. OWL은 rdf-sheme을 기반으로 한 지식표현 방법으로서[8,9], Racer에서 이 내용의 추론이 가능하다.

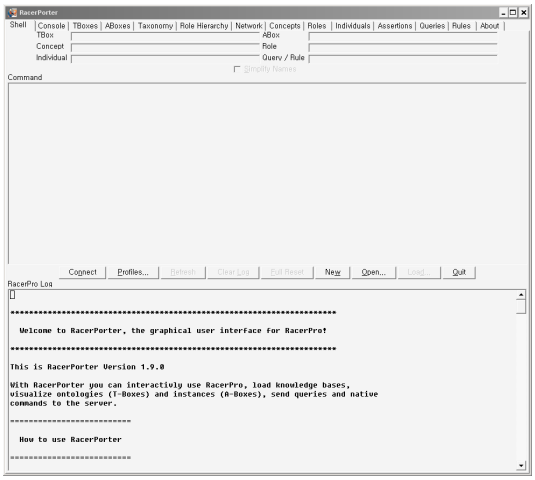
```
<?xml version="1.0"?>
<rdf:RDF
xmlns="http://www.owl-ontologies.com/Ontology1167624591.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.owl-ontologies.com/Ontology1167624591.owl">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:ID="Professional"/>
    <owl:DatatypeProperty rdf:ID="Name">
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="Major">
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>
    <Professional rdf:ID="prof1"/>
</rdf:RDF>
```

3.3 온톨로지 추론

온톨로지의 추론[10,11]은 Racer라는 추론기를 사용하였다. RACER[3]는 RDF/OWL로 표현된 지식을 추론 할 수 있는 프로그램이다. Racer는 2006년 현재 1.9버전까지 나와있으며, 이 또한 역시 자바로 이루어져 있어 플랫폼의 제약이 거의 없는 것이 특징이다. Racer는 Racer Pro 뿐 아니라 Ontology 구축을 디버깅 할 수 있는 Racer Poter가 같이 포함되어 있다.

Racer System은 RDF/OWL을 처리하는 semantic middleware로써 현재 W3C에서 표준으로 되어 있으며, RacerPro는 이 시스템의 커널(kernel) 프로그램으로써 현재 존재하는 메타 데이터(meta-data)를 제공하는 프로그램이다. Racer는 논리적 시스템을 표현하고 추론하는데 쓰일 수 있게 된다.

Racer Poter는 Racer에게 TCP/IP 또는 HTTP로 접속하여 현재 설정된 OWL 파일을 로드하여 추론 및 OWL 구조를 GUI(Graphic User Interface)로 볼 수 있다. Racer Pro는 보통 얼굴없는 서버(face-less back-end server)라 불리며, Racer Porter는 이것을 사용 하는 인터페이스(Interface)이다. 이 프로그램은 UI(User Interface)이기 때문에 이 프로그램만으로는 추론이 불가능 하다. 이 프로그램은 어느정도의 인터페이스를 제공 하긴 하지만, Racer Systems에서 추천하는 방법은 직접 API를 구현하라고 되어있다.



[그림 4] Racer Poter 실행화면

Racer는 자바로 이루어진 Racer를 사용할 수 있는 API(Application Interface)를 제공한다. 이는 Racer Poter와 같은 동작을 하는 API로써, 이 API를 이용하여 Racer를 사용한 어플리케이션 (Application)을 작성 할 수가 있다.

Racer 쿼리에서 데이터를 추출 할 수 있는 쿼리는 Retrieve 이다. Racer 쿼리는 Horn Logic 기반으로 만들어져 있어 쿼리의 중첩도 가능 한 것이 특징이 된다. 아래는 컴퓨터 공학과의 이벤트에서 신입생 오리엔테이션에 관한 질문에 대한 쿼리이다.

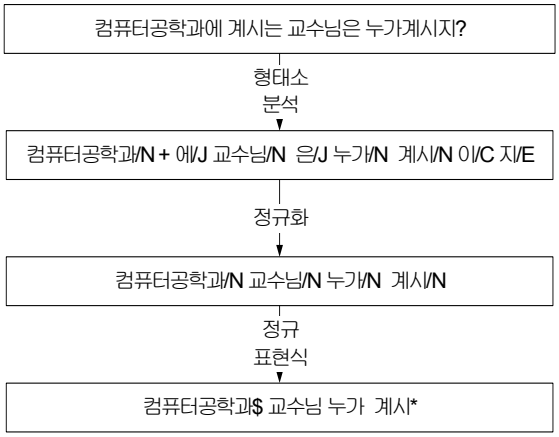
(Retrieve (?x ?y) (?x ?y |http://www.owl-ontologies.com/unnamed.owl#inverse_of_hasEventSinOT|))

쿼리문을 보게 되면 쿼리는 x에 대한 내용을 찾기 위해서 y에서 찾게 되는데, x와 y간의 관계는 inverse of hasEventSinOT이라는 뜻이다. 이렇게 쿼리를 Racer에게 주게 되면 Racer는 그 결과값을 추론하여 사용자에게 돌려주게 된다.

4. 어휘 의미 패턴 (LSP)

위에서 언급한 것과 같이 사용자의 입력은 항상 정해진 입력이 아니기 때문에 많은 같은 질의를 하더라도 다른 표현으로 입력 될 수 있게 된다. 다양한 입력을 더 작은 그룹으로 줄여 처리를 하면 간략하면서, 더욱 강력한 처리가 가능하게 되는데, 그 중 한 방법으로 어휘의 의미 패턴을 통해서 사용자의 입력을 그룹화[12] 시킬 수 있게 된다.

4.1 LSP 추출

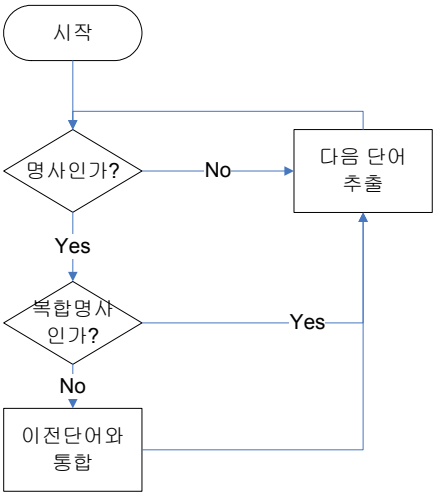


[그림 5] LSP 추출 과정

[그림 5]는 사용자입력을 LSP형태로 추출하고 추출된 내용을 사용해서 정규 표현식으로 만드는 과정을 도식화 시킨 것이다. LSP 추출 과정은 사용자가 질의한 문장을 먼저 형태소 분석기를 통하여 형태소 분석을 한다. 형태소를 분석하는 이유는 사용자 질의에서 중요한 키워드만을 추출 하기 위해 하는 전처리 과정이다. 이러한 전처리 과정을 통하여 LSP를 구성하는 코드들을 뺄 수 있다.

형태소 분석이 된 결과물을 가지고 패턴을 만드는데 전혀 필요가 없는 어휘들은 제거를 해주는 정규화 과정을 거치게 된다. 이러한 정규화 과정은 먼저 조사를 제거, 어미 등을 제거 한다. 그런 다음 복합 명사를 처리 하게 되는데 복합명사의 사용자의 입력은 띄어쓰기 하는 경우도 있고 모두 붙여쓰는 경우도 많기 때문에 복합 명사들을 처리해주는 과정이 필요하게 되는 것이다.

복합명사는 복합명사 사전을 구축하여 처리를 하였다. 복합 명사 사전의 개수는 약 정보검색기에 색인(indexing) 되어진 문서에서 추출하여 약 200개의 복합명사를 구축 하였다. 사용자 질의에서 “XXX의 YYY”같은 패턴도 복합 명사로 간주하여 처리를 하였다. 복합명사의 검색은 형태소 분석기에서 나타난 명사들을 조합하여 복합명사와 일치 되는지를 찾는 방법으로 구현하였다. [그림 6]는 복합명사를 어떠한 방법으로 처리 하였는지에 대한 순서도 이다.

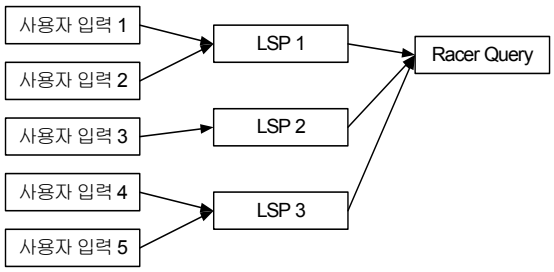


[그림 6] 복합명사 판단 과정

복합명사 처리 후, LSP 생성의 마지막 단계로 문장의 정규화 과정을 거치게 된다. 문장의 정규화 과정은, 불필요한 단어를 제거 하게 되는데, 제거되는 대상은 조사, 접속사 등과 같은 단어들이다. 정규화 과정 이후에는 정규 표현식으로 만들어 패턴 매칭의 정보로 사용 하게 된다.

4.2 패턴 매칭

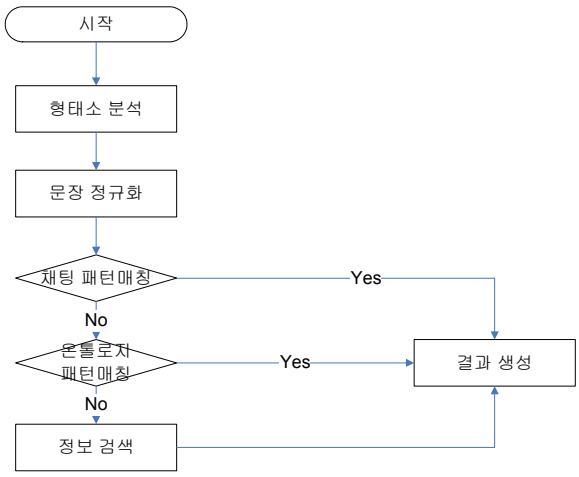
[그림 3]에서 보이는 것처럼 하나의 데이터베이스 질의문은 여러 개의 LSP형태와 대응되고, 각 LSP 형태는 1개 이상의 사용자 입력 문장과 대응되게 된다.



[그림 7] 사용자 질의 입력과 LSP와 Racer Query의 대응 관계

[그림 7]에서 보이는 것처럼, 사용자 질의를 알맞은 패턴으로 그룹화하여 처리를 하기 때문에 비슷한 문장을 효과적으로 처리할 수 있다.

패턴매칭을 하는데에는 2가지의 큰 그룹이 있다. 하나는 온톨로지 추론이 필요한 그룹이고, 나머지 하나는 채팅그룹이다. 채팅 그룹이 필요한 이유는 사용자 질의가 문장의 형태로 들어오기 때문에 사용자는 입력을 시스템에 관한 질문을 할 수도 있게 된다. 이러한 것은 온톨로지로 구축하는 것이 아니라 채팅으로 분류하여 처리를 해야 한다. 예를 들어 "안녕"이라는 질문을 했을 경우 온톨로지에는 이 정보가 없게 된다. 하지만 채팅에는 이러한 처리를 할 수 있게 함으로써 보다 친근한 시스템이 될 수 있다.



[그림 8] 패턴 매칭 순서

[그림 8]은 온톨로지와 채팅에 관한 매칭 순서를 나타내었다. [그림 8]에서 처럼 문장이 들어오게 되면 문장의 LSP를 추출하고, 그것을 채팅, 온톨로지 순으로 검색을 거치게 된다.

5. 온톨로지로 표현된 시스템의 이점

5.1 다중 질문의 효율적 처리

온톨로지를 사용함으로써의 이점은 사용자가 입력한 질문이 특정한 범위의 내용이 아니라 넓은 의미의 질문을 하는 경우에 크나큰 이점이 존재 한다. 이러한 이점은 데이터베이스 쿼리의 경우에는 특정한 값만을 추출하여 사용 하기 때문에 질문의 답변은 한정적이게 된다. 하지만 온톨로지를 사용하여 하는 경우에는 더 작은 지식의 구축으로 데이터베이스보다 훨씬 넓은 질의의 처리가 가능하게 된다.

예를 들어 질문이 "컴퓨터공학과에 대해 말해줘"라는 질문이 들어왔다고 한다면, 데이터베이스에서는 이것에 대한 답변을 찾을 수가 없게 된다. 하지만 온톨로지의 경우 컴퓨터 공학과에 연결된 관계를 모두 연결하여 질문의 답변이 가능하다. 예를 든 질문의 답변으로 아래의 예처럼 나타날 수가 있게 되는 것이다.

"*컴퓨터공학과의 교수는 P1, P2, P3가 있고, 행사는 E1, E2. E3등이 존재하며 동아리는 D1, D2, D3가 있습니다.*"

이러한 지식을 일반 데이터베이스에서 찾아내려면 개발자가 일일이 이것에 관한 쿼리문(query)을 다 만들어 주어야 하며, 이런 쿼리문은 하나로 표현 할 수가 없게 된다. 하지만 온톨로지로 하게 되면 단순히 all-individuals 라는 쿼리만으로 이러한 내용이 가능 하게 된다. all-individuals는 모든 개체들(individual)을 리턴하는 쿼리문으로 이것을 사용하면 단 하나의 쿼리로 일반 데이터베이스에서는 쉽지 않은 질의 처리를 간단하게 처리를 할 수가 있게 된다.

5.2 질의 확장

온톨로지의 또 다른 이점은 질의를 쉽게 확장이 가능하다는 것이다. 온톨로지는 지식이 관련된 개념에 대한 관계로 표현되어져 있어 기존의 데이터베이스와는 달리 질의 확장을

위한 새로운 정보 생성 없이 질의 확장이 가능하다. 이렇게 얻어진 질의를 사용하여 구축된 정보검색기에 간단한 실험을 하였다. 실험에 쓰여진 데이터는 동아대학교 컴퓨터공학과 자유게시판의 내용이며, 정확률의 계산은 문서 수준 정확률의 R-정확률을 사용 하였다.

R-정확률은 상위 R개의 문서들의 포함된 문서들의 비율이다. R-정확률 계산식[13]은 아래와 같다.

$$R-\text{정확률} = \frac{\text{상위R개의문서들에포함된적합문서수}}{R} \quad (1)$$

5.3 질의 확장 실험 결과

다음은 구축된 시스템에서 정보검색 질의 확장한 예이다. 결과는 R-정확률을 사용하였다.

[표 1] 질의 확장의 실험 결과의 예

구분	*내용*	
확장전쿼리	컴퓨터 공학과 행사	54.54%
확장후쿼리	스승의날 오리엔테이션 하나제 MT	62.5%
성능향상		+7.96%
확장전 쿼리	교수	20%
확장후 쿼리	김○○, 박○○, 이○○, 허○○, 권○○, 고○○	42.85%
성능향상		+22.85%

[표 1]에서 보여지는 것처럼, 온톨로지로 부터의 질의확장을 사용한 결과 더 좋은 정확률을 얻을 수 있었다.

6. 결론 및 향후과제

본 논문에서 제안하는 정보검색기는 사용자 질의를 자연어의 문장 형태로 입력을 받아 질의 패턴을 만들어 내고, 온톨로지 기술과 어휘의미패턴을 사용해서 질의에 대한 정답을 효과적으로 제공하고 필요할 경우 정보검색기의 질의를 확장하여 정보검색기의 정확률을 올린다. 사용자 질의가 일반 질의와 달리 사용자가 직접 쓰는 말을 그대로 입력하여 처리하여야 하기 때문에 까다로운 작업을 많이 해야 하지만 사용자가 입력하는 문장을 패턴화시켜 그에 맞는 패턴을 찾아 내고 패턴 그룹화를 통해 처리함으로써 보다 효과적으로 사용자 질의를 처리 할 수 있는 방법을 본 논문에서 제시하였다.

이러한 온톨로지를 이용하여 지식을 표현하여 사용하면서 얻어지는 이점은 두 가지로 압축 할 수가 있었다. 하나는 모호한 질문에 대한 데이터베이스 쿼리수의 감소와 다른 하나는 검색기의 질의의 개수를 자동으로 확장이 가능 한 것이었다.

향후 연구로는 현재 시스템의 경우 하나의 사이트를 일반 사용자에게 소개하는 것에 국한 시켰지만, 여러 사이트 간의 관계를 이용하여 온톨로지를 구성하여 정보검색을 한다면 더욱 효율적인 정보검색을 수행할 수 있을 것이다.

감사의 글

이 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2006-331-D00536)

참고문헌

[1] C. Golbreich, “Combining Rule and Ontology Reasoners for the Semantic Web,” LNCS, No. 3323, pp.6-22, Springer, 2004.

[2] P.S. Jacobs, G.R. Krupka, and L.F. Rau, “Lexico-Semantic Pattern matching as a companion to parsing in text understanding,” Fourth DARPA Speech and Natural Language Workshop, pp. 337-342, 1991.

[3] 한경민, “한국어 DB 질의 처리를 위한 은닉 마르코프 모델의 적용”, 서강대학교 석사 학위 논문, 2004.

[4] 오티스 고스포드네티치, 에릭 해쳐, “Lucene in Action”, 에이콘. 2005.

[5] http://www.racer-systems.com/racerpro/index.phtml, “What is RacerPro?,” Racer Systems GmbH & Co. KG, 2005.

[6] http://www.racer-systems.com/products/porter.phtml, “Porter, the native, graphical user interface for RacerPro,” Racer Systems GmbH & Co. KG, 2005.

[7] http://protege.stanford.edu/plugins/owl/ “What is protégé-owl?”, Stanford Medical Informatics. 2006.

[8] http://www.racer-systems.com/, “Racer User Guide,” Racer Systems GmbH & Co. KG, 2005.

[9] http://www.racer-systems.com/, “Racer Reference Manual,” Racer Systems GmbH & Co. KG 2005.

[10] C. Golbreich, “Combining Rule and Ontology Reasoners for the Semantic Web”, In conjunction with the International Semantic Web Conference, 2004.

[11] C. Golbreich, A.Imai, “Combining SWRL rules and OWL ontologies with Protégé OWL Plugin, Jess, and Racer”, 7th International Protégé Conference, 2004.

[12] G.G. Lee, J. Seo, S. Lee, H. Jung, B. Cho, C. Lee, B. Kwack, J. Cha, D. Kim, J. An, H. Kim, and K. Kim, “SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP,” Proc. of the Tenth Text REtrieval Conference, 2001.

[13] C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 33-40. ACM Press, 2000.

[14] Baeza-Yates, Ribeiro-Neto, “최신 정보 검색론 (Modern Information Retrieval)”, 홍릉과학출판사, 1999.