SPARQL 기반의 질의응답 시스템 설계

안혁주○, 이성희, 김학수
강원대학교 컴퓨터정보통신공학과
zingiskan12@kangwon.ac.kr, tjdgml0820@naver.com, nlpdrkim@kangwon.ac.kr

Design of a Question-answering System Based on SPARQL

HyeokJu Ahn○, SungHee Lee, HarkSoo Kim
Kangwon National University Computer and Communication Engineering

요 약

사용자가 질의한 내용에 대한 결과를 찾기 위해 본 논문은 DBPedia에서 제공해주는 트리플 구조를 TDB에 저장하고, 사용자 질의 문장에서 트리플을 찾은 뒤 해당 문장의 규칙을 추론하여 SPARQL 쿼리를 생성한 뒤, 마지막으로 Fuseki를 이용해 결과를 출력하는 Q&A시스템을 제안한다. SPARQL 쿼리를 생성함에 있어 질의의 정답을 찾아내는 타겟이 있다는 점과 한국어의 조사와 부사부분에서 쿼리가 변형될 수 있다는 점을 통해 유동적인 쿼리를 생성한다. 그리고 DBPedia에 없는 단어가 질의에서 나타날 수 있기 때문에 이를 정제해주는 작업 또한 필요하다. 한국어는 어절순서가 고정적이지 않다는 점, 조사, 부사에 의해 문장의 의미가 변형되는 또 다른 부분을 파악하여 앞으로 시스템을 개발함에 있어 정확률을 상승시킬 예정이다.

주제어: 질의응답 시스템, 트리플, TDB, Fuseki, SPARQL

1. 서론

최근 Natural Language Q&A 시스템은 Q&A시스템의 새로운 지표로 자리매김하고 있으며 해외에서도 관심이 많은 분야 중 하나이다. 질문에 대해 단순히 키워드를 찾아 출력하는 예전 Q&A시스템과 달리 Natural Language Q&A 시스템은 사용자의 질문에 따라 유동적으로 답변을 해줄 수 있기 때문에 그 활용가치가 높다. 이러한 Q&A시스템을 개발하는데 최근 가장 많이 활용하는 방법은 주어, 술어, 목적어 정보를 가지는 트리플(Subject, Predicate, Object)구조를 이용하는 것으로, 이 구조를 통해 많은 질의에 대해 유동적으로 답변을 할 수 있다. 이러한 트리플 구조를 채용하여 쿼리문법으로 표현한 것이 바로 SPARQL[1]이다.

해외에서도 트리플 구조와 SPARQL을 활용한 여러 연구들이 이루어지고 있다. 본 논문에서 기반으로 하는 연구는 문장에 대해 트리플을 추출한 뒤 온톨로지 정보와 추출한 트리플을 서로 연결하여 SPARQL 쿼리를 찾아내는 연구[2]와 두 번의 SPARQL 쿼리를 던져 이미 존재하는 데이터베이스 내에서 정보를 찾는 연구[3], 마지막으로 Jena RDF와 SPARQL을 이용한 시멘틱웹을 구축하는 연구[4]이다. 위 세 논문을 기반으로 본 논문에서는 Triple Store TDB[5]를 이용하여 DBPedia[6]에서 제공하는 트리플을 저장하고, 사용자가 입력한 한국어 질의에서 트리플을 추출한 뒤, 문장의 규칙을 추론하여 SPARQL 쿼리를 생성한다. 마지막으로 Fuseki[7]를 이용하여 SPARQL 쿼리에 대한 결과를 출력하는 연구를 제안한다.

* 이 논문은 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2013R1A1A4A01005074) 또한 본 연구는 LG전자 산학연구용역 과제의 지원을 받아 수행되었음

2. 시스템 구성 및 설계

2.1 시스템 순서도

본 논문에서 설계한 시스템의 순서도는 [그림 1]과 같다. 시스템은 트리플 DB Store와 실행시스템 두 부분으로 구성된다. 트리플 DB Store는 DBPedia의 Triple 파일을 TDB에 단순히 저장하는 것으로 SPARQL 쿼리를 생성하는데 쓰인다. 실행 시스템은 사용자가 입력한 문장에서 트리플을 모두 추출한 뒤, 문장의 형식과 질의 기준에 따른 규칙을 추론한다. 추론한 규칙으로 SPARQL 쿼리를 생성하는데 그 전에 TDB를 이용하여 트리플의 URI를 추출해 낸다. 사용자가 입력한 문장의 트리플이 TDB에서 찾을 수 없는 경우를 대비하여 유의어 사전을 사용한다. 마지막으로 TDB와 연결된 Fuseki를 이용하여 생성된 SPARQL쿼리를 실행한 뒤 그 결과를 출력한다.

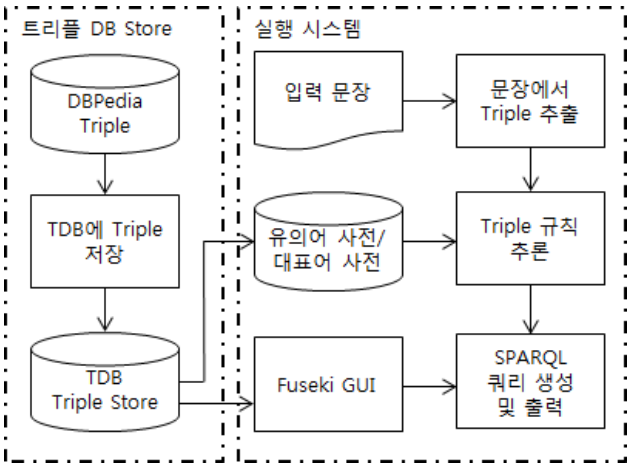

그림 1 시스템 순서도 디자인

2.2 문장 트리플 추출과 규칙 추론

앞서 설명한 바와 같이 문장에서 추출할 트리플은 Subject, Predicate, Object 구조로 이루어져 있다. 이 트리플은 한 문장에서 하나만 추출되는 것이 아니라 문장의 구조에 따라 여러 개의 트리플이 추출될 수 있다. 트리플은 어떤 문장에 대해 구문분석기를 이용하여 어절의 전후관계를 파악한 뒤, 동사구를 트리플의 Predecate로 간주하여 트리플을 추출하는 것이 기본이다. 그러나 한국어는 어절의 순서가 바뀔 가능성이 있고, 동사구를 생략한 표현이 발생할 수 있기 때문에 이에 대한 별도의 처리가 필요하다. [표 1]은 어떤 문장에서 트리플이 추출되는 예를 보여준다.

표 1 문장에서 트리플 추출 예

문장1	과학자 중에서 국적이 한국이거나 일본인 인물은?		
트리플1	과학자	국적	한국
트리플2	과학자	국적	일본
문장2	미국에서 태어나고 키가 가장 큰 농구선수와 키가 가장 작은 농구선수의 이름은?		
트리플1	농구선수	태어남(출생지)	미국
트리플2	농구선수	키	가장 큰
트리플3	농구선수	키	가장 작은

[표 1]과 같이 문장에서 추출된 트리플은 TDB에 저장된 URI를 추출할 수 있는 SPARQL 쿼리를 생성할 수 있도록 문장규칙을 추론해야한다.

문장에 대해 나올 수 있는 정답이 무엇인지를 추론하기 위해서 우선 질문에서 얻고자 하는 타겟이 필요하다. 한국어에서는 대체로 Subject가 타겟이지만 다른 정보, 예를 들면 문장2와 같은 농구선수의 이름과 키까지 확인해야 되는 경우는 추가적으로 타겟을 지목한다. 문장1에서의 '~거나(OR)', 문장2에서의 '~나고(AND)'와 같이 한국어에서는 조사부분에서 문장의 특징을 판별할 수 있는 단서가 나올 수 있기 때문에 각각의 트리플을 연관시켜주는 조사를 찾는 것 또한 중요하다. 또한 부사에서도 문장의 특징을 판별할 수 있는데, 문장2에서는 '가장'이라는 키를 꾸며주는 부사가 있으므로 추출된 목록 중에서 가장 우선순위인 값들을 추출해야 한다. 따라서 추출된 농구선수의 오름차순과 내림차순 목록이 필요한 것을 알 수 있다.

2.3 대표어 사전/Edit Distance

질의에서 나온 단어가 DBPedia에서 그대로 나오지 않을 가능성이 있기 때문에 URI가 추출되지 않을 수 있다. 이를 대비하여 트리플로 추출된 단어를 정제하는 대표어 사전이 필요하다.

대표어 사전은 단어목록들의 대표어를 추출해줄 수 있는 사전과 이 대표어들을 DBPedia에서 대표되는 속성명으로 바꿔주는 두 가지 사전이 필요하다. [표 2]는 DBPedia에서 쓰이는 속성 명 <wasBornIn>을 기준으로 추출될 수 있는 목록이다.

표 2 <wasBornIn>에서 나올 수 있는 대표어 사전 내용

DBPedia 속성명 대표어 사전	wasBornIn	
추출되는 속성명	출생지, 고향	
단어 대표어 사전	출생지	고향
단어 목록	태어남, 출생, 나오다, ...	

또한 단순히 원하는 대표내용만이 출력되는 것이 아닌 세부내용까지 출력되는 경우가 있다. 예를 들면 '출생지'라는 속성에서 단순히 국가만이 출력되는 것이 아닌 세부 주소까지 출력되는 경우일 것이다. 이를 대비하여 TDB에서 트리플을 확인할 때, 목적어에 포함되는지 여부를 확인하는 과정이 필요하다.

사용자가 질의를 입력할 경우, 입력과정에서 실수할 가능성이 있다. 예를 들면 '과학자'라는 단어를 '과학지' 등으로 잘못 쓴 경우인데, 이를 보완하기 위해 Edit Distance기법을 사용한다.

2.4 SPARQL 쿼리 생성

본 논문에서는 [표 1]에서 보인 문장을 토대로 SPARQL 쿼리를 생성하는 예를 설명할 것이다. SPARQL은 트리플 구조로 형성되어 있으며, 정답을 추출하기 위한 일련의 과정이 필요하다.

```
SELECT ?과학자
WHERE{
   ?과학자   rdf:type  :Scientist  .
   {
        {?과학자  :국적  :한국  .}
     UNION
        {?과학자  :국적  :일본  .}
   }
}
```

그림 2 [표 1] 문장1의 SPARQL 쿼리

[그림 2]는 문장1의 SPARQL 쿼리이다. SELECT에 해당 문장에서 타겟으로 지정한 '과학자'를 넣어 질의에 대한 응답이 과학자에 대한 목록으로 추출될 수 있도록 설정한다. 문장1에서 '~거나'라는 문장의 특징을 찾았기 때문에 국적이 한국, 일본인 모든 과학자 목록을 추출해야 한다. 이를 위해 OR을 표현해 주는 UNION이라는 쿼리 문법을 통해 결과를 출력한다.

```
SELECT ?농구선수 ?키
WHERE{
   ?농구선수   rdf:type   :BasketballPlayer   .
   ?농구선수   :출생지   :미국   .
   ?농구선수   :신장   ?키   .
}
ORDER BY ASC(?키)
LIMIT   1
ORDER BY DESC(?키)
LIMIT   1
```

그림 3 [표 1] 문장2의 SPARQL 쿼리

[그림 3]은 문장2의 SPARQL 쿼리이다. 키에 따른 농구선수 정답을 출력하는 것이므로 타겟은 '농구선수', '키' 이다. 문장2에서 '~나고' 라는 문장의 특징을 찾았기 때문에 '?농구선수' 에 정보를 추가해주어 AND를 표현한다. [표 1]에서는 트리플이 3개가 추출되었지만 그 중 2개의 트리플은 '미국에서 태어난 농구선수' 라는 공통정보 내에서 찾기 때문에 오름차순과 내림차순으로 목록을 추출한 뒤, 제한을 1로 두어 결과를 출력한다.

본 논문에서는 SPARQL쿼리를 생성하여 그것을 출력하기 위한 응용프로그램으로 Fuseki를 사용하는 것을 제안한다. 추후에 SPARQL이 제대로 되는지를 확인하기 위해 Fuseki를 TDB에 연결하여 프로세스에서 제공하는 GUI를 통해 결과를 출력할 것이다.

3. 결론 및 향후 연구

본 논문에서는 SPARQL을 이용한 Q&A시스템을 개발하기 위한 과정에 대해 설명하였고, 이를 구현하기 위해 DBPedia 트리플을 저장하기 위한 TDB, 사용자가 질의한 문장을 SPARQL질의어로 만들고, 그 결과를 출력할 수 있게 도와주는 Fuseki를 사용할 것을 제안하였다. 현재 본 논문에서 작성한 연구를 통해 개발 중인 Q&A시스템에 대한 정확률 상승을 기대하고 있다. 현재 개발함에 있어 가장 큰 걸림돌은 문장에 대한 규칙을 추론하는 것이다. 영어와 달리 한국어는 단어별로 이루어져 있지 않고 각각의 형태소가 결합되어 한 어절을 이루고, 조사와 부사에 따라 문장의 의미가 바뀔 수 있기 때문에 규칙을 추론하기 어려운 점이 있다. 따라서 향후 연구로 질문 문장의 여러 가지 변수를 고려하는 방법을 찾아 시스템을 개발함에 있어 정확률을 상승시킬 예정이다.

참고문헌

[1] ARQ - A SPARQL Processor for Jena, "http://jena.apache.org/documentation/query/index.html"

[2] Chong Wang, Miao Xiong, Qi Zhou, Yong Yu, "PANTO:A Portable Natural Language Interface to Ontologies", The Semantic Web:Research and Applications Lecture Notes in Computer Science Volume-4519, pp. 473-487, 2007.

[3] JD Kim, KB Cohen, "Natural Language Query Processing for SPARQL Generation - a Prototype System for SNOMEDCT", BioLINK - biolinksig.org, 2013

[4] Michael Grobe, "RDF, Jena, SparQL and the "Semantic Web" ", 37th annual ACM SIGUCCS fall conference : communication and collaboration, pp. 131-138, 2009

[5] TDB, "http://jena.apache.org/documentation/tdb/index .html"

[6] DBPedia Downloads, "http://wiki.dbpedia.org/Downloads39"

[7] Fuseki, "http://jena.apache.org/documentation/serving _data/index.html"