RESTful 웹 서비스에서 리소스 발견 방법

이용주
경북대학교 과학기술대학 컴퓨터정보학부
e-mail:yongju@knu.ac.kr

Resource Discovery Method in RESTful Web Services

Yong-Ju Lee
School of Computer Information, Kyungpook National University

요 약

근래에 OpenAPI의 구현 형태는 기존의 SOAP 기반 구현 방식에서 비교적 간단하고 가벼운 REST 방식으로 바뀌고 있다. 이러한 결과로 웹상에 이용 가능한 RESTful 웹 서비스들의 수가 급격하게 증가됨에 따라 적합한 리소스를 찾는 것은 매우 중요한 이슈로 대두되었다. 본 논문에서는 RESTful 웹 서비스를 개발할 때 생성되는 WADL 문서를 가지고 리소스를 효율적으로 발견할 수 있는 일련의 다단계 매칭 방법을 제안한다. 제안된 방법은 168개의 RESTful 웹 서비스 집합에 대한 실험을 수행하여 그 성능의 우수함을 보인다.

1. 서론

최근 웹 2.0의 등장과 함께 OpenAPI(Application Program Interface)와 매쉬업(mashup)이 발전되면서 기존의 SOAP 기반 웹 서비스에 비해 RESTful 웹 서비스의 활용이 크게 증가하고 있다[1]. 구글, 아마존, 야후, 이베이, 네이버, 그리고 다음과 같은 기업에서는 웹 2.0의 새로운 패러다임에 발맞추어 자사의 정보 자원을 OpenAPI를 통해 외부 사용자에게 적극적으로 개방하고 있는 추세이며, OpenAPI 구현 형태도 기존의 SOAP 기반 구현 방식으로부터 비교적 간단하고 가벼운 REST(REpresentation State Transfer)[2] 방식으로 바꾸고 있다. 최근 동향을 살펴보면, OpenAPI와 매쉬업을 위한 대표적인 포털 사이트인 ProgrammableWeb[3]에서 제공되는 OpenAPI의 약 72%(2/3)가 REST 방식이고 18%가 SOAP 방식이다.

이러한 결과로 웹상에 이용 가능한 RESTful 웹 서비스의 수가 기하급수적으로 증가됨에 따라 사용자가 적합한 리소스를 발견하는 것이 매우 중요한 이슈로 대두되고 있다. 그러나 기존의 키워드 기반 검색 방법은 이미 여러 연구에서 밝혀진 바와 같이 나쁜 재현율과 나쁜 정확률 때문에 문제가 많다. 이러한 키워드 기반 검색 방법의 한계를 극복하기 위하여 시맨틱 정보를 이용한 온톨로지(ontology) 활용 방법이 있을 수 있다[4, 5]. 그렇지만 이러한 시맨틱 온톨로지 활용 방법은 대부분 SOAP 기반 웹 서비스를 위한 방법이었고 RESTful 웹 서비스를 위해서는 이들이 잘 맞지 않는다. 왜냐하면, SOAP 기반 웹 서비스에서는 다양한 오퍼레이션들에 대한 시맨틱 처리가 중요한 반면에, RESTful 웹 서비스에서는 이들 오퍼레이션 대신에 체계적으로 구성된 URI에 대한 HTTP 기본 메소드만 수행되기 때문이다.

예를 들면, SOAP 기반 웹 서비스에서는 주문・제품 정보 등을 생성, 검색, 변경하는 오퍼레이션들이 각기 존재하여 필요한 기능을 수행할 때 해당 오퍼레이션(예, getOrder( ))을 호출하는 방식으로 일반적인 프로그래밍 개념과 동일한 반면에, RESTful 웹 서비스에서는 주문 리스트(/order), 특정 id에 대한 주문 정보(/order/{id}), 제품 리스트(/product), 특정 id 제품 정보(/product/{id}) 등을 모두 리소스로 정의하고, 각 리소스에 URI를 할당한 후 그 URI에 대해(예, http://korea.com/order) GET 또는 POST 메소드를 수행한다.

본 논문에서는 RESTful 웹 서비스를 개발할 때 생성되는 WADL(Web Application Description Language) 문서를 가지고 RESTful 웹 서비스를 위한 유사 리소스 발견 방법을 제안한다. 제안된 방법은 구문 분석(syntactic analysis) 방법과 클러스터링(clustering) 방법을 혼합 사용하여 보다 지능적인 리소스 매칭을 수행한다.

본 논문의 구성은 다음과 같다. 2장에서 RESTful 웹 서비스의 기본 개념과 기술언어를 간단히 기술하고. 3장에서 RESTful 웹 서비스 리소스 발견 방법을 제안한다. 4장에서 실험 분석을 수행하고 5장에서 결론을 내린다.

2. RESTful 웹 서비스

REST는 웹의 창시자 중 한 사람인 Roy Fielding의 박사학위 논문[2]에 의해 소개되었다. 그는 현재의 웹 아키텍처가 웹이 지닌 본래의 설계 우수성을 충분히 활용하

이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(No. 2010-0008303).

지 못하고 있다고 판단하고, 웹의 장점을 최대한 활용할 수 있는 네트워크 기반의 아키텍처를 제안했는데 그것이 바로 REST다. 이런 REST 아키텍처 스타일에 따라 정의되고 이용되는 서비스나 응용을 RESTful 웹 서비스라 한다. RESTful 웹 서비스의 기본 개념은 다음과 같다.

- 리소스: RESTful 웹 서비스의 가장 큰 특징 중의 하나는 모든 대상을 리소스(resource), 즉, 자원으로 표현한다는 것이다. 이 리소스는 HTTP URI(Uniform Resource Identifier)에 의해 표현되며, user, product, order 등과 같이 직관적으로 리소스를 인식할 수 있는 명사형의 단어들로 설계되고, /user/1234와 같이 계층 구조로 구성된다.
- HTTP 메소드: REST 구조에서의 리소스는 HTTP의 기본 메소드(method)인 POST, GET, PUT, DELETE만으로 접근할 수 있다. 리소스에 접근하기 위한 이러한 4개의 HTTP 메소드는 일반 CRUD(Create, Read, Update, Delete) 오퍼레이션에 각각 대응될 수 있다.
- 다양한 표현: 하나의 리소스는 다양한 표현(representation)을 가질 수 있는데, 이는 XML, JSON, XHTML 등이 가능하며 클라이언트에서 원하는 형식을 서버에게 요청할 수 있다. 표현은 다른 자원을 참조하는 링크(link)를 포함할 수 있으며, 만일 클라이언트가 링크를 따라 다른 자원으로 이동한다면 이는 그 상태(state)를 변경(transfer)하는 것이다. 이 개념은 HATEOAS(Hypermedia As The Engine Of Application State)라는 용어로 표현된다.
- 스테이트리스: 스테이트리스(stateless)란 웹 서비스 제공 서버 측에서 클라이언트의 상태(state) 정보를 저장, 관리하지 않는 것을 의미한다. 즉, 클라이언트가 HTTP 요청(request)을 할 때 서버에 그 요청을 수행할 수 있는 모든 정보를 주어야 하며 이전의 요청에 의존해서는 안 된다. 이런 특성은 상태를 저장 하지 않으므로 확장성이 좋아지고 캐쉬(cache)하기에 적합한 구조를 만든다.

이상과 같이 RESTful 웹 서비스는 리소스의 URI만 알면 SOAP 기반 웹 서비스와 같은 부가적인 전송 레이어 없이 HTTP 프로토콜만으로 접근 가능한 아주 간단한 서비스라 할 수 있다. 이러한 단순 명료한 접근 방식 때문에 구글, 야후, 트위터 등에서 제공하는 대부분의 웹 2.0 OpenAPI들이 RESTful 웹 서비스로 작성되어 있으며, 이는 위젯(widget)을 이용한 매쉬업을 활성화시킨 원동력이 되었다. 또한, 기존에 제공되던 SOAP 기반 웹 서비스 조차도 RESTful 웹 서비스를 추가 개발하여 동시에 제공하는 추세이다[6].

SOAP 기반 웹 서비스에 비해 가볍고 구현하기 쉬운 RESTful 웹 서비스가 많은 주목을 받고 있음에 따라, RESTful 웹 서비스의 인터페이스를 기술하기 위한 다양한 방법들이 제안되고 있다. 현재 RESTful 웹 서비스를 기술하기 위한 여러 가지 언어들이 제안되고 있으나 아직까지 주류는 형성되지 않은 상황이다. 기존의 WSDL에서도 2.0 버전에서는 RESTful 웹 서비스를 기술할 수 있는 HTTP 바인딩(binding) 확장 스펙이 제안 되었지만 너무 복잡하다는 이유로 많이 쓰이지는 못하고 있다. 이에 썬마이크로시스템은 간략하면서도 범용성이 뛰어난 WADL[7]을 발표하게 되었고, 간단하다는 장점 때문에 개발자들 사이에서 WADL의 사용은 점차 증가되고 있는 실정이다.

WADL은 HTTP 기반 웹 애플리케이션(예를 들면, RESTful 웹 서비스)의 기계 판독이 가능한 기술을 제공하는 XML 기반의 문서 형식이다. WADL의 목적은 보다 쉽게 웹 2.0 형태의 애플리케이션들을 생성하고 동적 서비스 생성 및 관리 방법을 제공하기 위하여, 인터넷 상의 서비스들을 기계가 처리할 수 있는 방법으로 기술할 수 있는 방안을 제공하는 것이다. WADL은 현존하는 웹 구조에 기반 한 애플리케이션들을 위하여 고안되었다. 즉, WSDL처럼 플랫폼과 언어에 독립적이며 웹 브라우저의 기본적인 사용 외에 애플리케이션의 재사용을 촉진시키는 것을 목표로 한다. 또한 WADL에서는 서비스에 의해 제공되는 자원들과 그들 사이의 관계를 모델링할 수 있다.

```
<resources base="http://news.search.yahoo.com/">
 <resource path="newsSearch">
  <method name="GET" id="search">
   <request>
    <param name="appId" type="xsd:string"
           style="query" required="true"/>
    <param name="query" type="xsd:string"
           style="query" required="true"/>
    <param name="type" style="query" default="all">
           <option value="all"/>
           <option value="any"/>
           <option value="phrase"/>
    </param>
    <param name="results" style="query"
           type="xsd:int" default="10"/>
    <param name="start" style="query" type="xsd:int"
           default="1"/>
    <param name="sort" style="query" default="rank">
           <option value="rank"/>
           <option value="date"/>
    </param>
    <param name="language" style="query"
           type="xsd:string"/>
   </request>
   <response status="200">
    <representation mediaType="application/xml"
           element="yn:ResultSet"/>
   </response>
   <response status="400">
    <representation mediaType="application/xml"
           element="ya:Error"/>
   </response>
  </method>
 </resource>
</resources>
```

(그림 1) WADL의 예

WADL에서 서비스는 resource 엘리멘트들로 기술되며, 이들 각각에는 request와 response를 기술하는 method 엘리멘터가 있다. request 엘리멘트에는 어떻게 입력을 표현할 것인가를 기술하고 있고, response 엘리멘트에는 서비

스 결과의 representation과 상태 정보를 기술하고 있다. 그리고 request와 response에는 매개변수를 나타내는 param 엘리멘트들이 있을 수 있다. 이러한 WADL 문서의 예는 (그림 1)과 같으며, 전체적으로 resource – method – request/response 트리 구조로 되어있다. REST 개발 툴(예, REST Describe & Compile[8])을 사용하면 이러한 WADL 파일은 쉽게 생성할 수 있다.

3. 리소스 발견 방법

사용자들이 원하는 RESTful 웹 서비스에 대한 검색 유형은 다음과 같이 두 종류로 요약될 수 있으나, 본 논문에서는 선행 연구로써 RESTful 웹 서비스 리소스 발견 문제에 초점을 맞춘다.

- 리소스 발견: 하나의 리소스가 주어졌을 때 이와 유사한 리소스를 찾는다. 직관적으로 비슷한 입력과 비슷한 출력을 가지고 있고 이들 입출력 사이의 관계가 비슷하다면 두 리소스는 유사하다고 할 수 있다.
- 리소스 조합: 하나의 리소스로는 다양하고 복잡한 사용자 요구사항을 충분히 만족시켜줄 수 없을 때 리소스 조합을 통해 새롭고 유용한 솔루션을 얻는다. 예를 들면, 여행 예약 서비스는 항공기, 숙박, 그리고 관광 등 다수의 서비스들이 적절히 결합되어야 한다.

RESTful 웹 서비스의 리소스를 발견하기 위해 본 절에서는 리소스의 유사도가 어떻게 측정되는지 기술한다. 하나의 리소스 템플릿이 주어졌을 때, 이 템플릿과 저장소 내의 임의의 리소스와의 유사도는 path 속성과 입출력 매개변수에 깊은 관련이 있다. 질의 템플릿 T는 다음과 같이 4개의 튜플(tuple)로 표현될 수 있다.

$$T = < method, path, request, response >$$

여기서, method는 {POST, GET, PUT, DELETE} 중 하나이고, path는 리소스 이름, request와 response는 입출력 항목을 표시하고 있다.

본 연구에서 유사 리소스 발견 과정은 다단계 매칭 방법에 의해 수행된다. 각 매칭 단계에서 질의 템플릿과 저장소 내의 RESTful 웹 서비스 사이의 일치 여부가 판단되는데, 이를 위해 먼저 사용자로부터 작성된 질의 템플릿을 어떠한 순서로 수행할지에 대한 질의 계획을 세울 필요가 있다. 질의 계획을 세울 때는 질의 처리의 효율을 고려해야 하는데, 매칭 요소의 특성상 매칭 속도가 빠른 항목이 있고 매칭 속도가 느린 항목이 있으며, 코드별 1차 필터링(filtering) 단계와 제약조건에 따른 2차 정제(refinement) 단계가 있기 때문이다. 따라서 계산량이 많은 매개변수 유사도 매칭은 다른 항목들보다 늦게 처리하고, method나 path 값의 경우 적은 계산량으로 높은 질의 처리 효율을 가지고 있으므로 먼저 처리한다.

method 매칭: REST에서 HTTP 메소드는 중심 역할을 수행한다. 이는 POST, GET, PUT, DELETE 4개의 메소드만 사용하므로 코드 일치 방법을 사용하여 쉽게 탐색될 수 있다. 즉, 같은 메소드에 속하는 서비스들만 빠르고 쉽게 필터링하여 다음 매칭 단계가 효율적으로 수행될 수 있도록 탐색 범위를 한정시킨다.

path 속성 매칭: 리소스 path 속성은 텍스트로 구성되어 있으며, 이런 텍스트에 대해 완전 일치 방법을 사용하면 너무 제한된 탐색이 이루어질 수 있기 때문에 좀 더 유연한 매칭이 필요하다. 본 연구에서는 구문 분석 방법[9]을 사용하여 시맨틱 매칭 유사도를 구한다. 본 유사도를 측정하기에 앞서 정확성을 향상시키기 위해 path 속성 값들에 대해 다음과 같은 전처리 과정을 수행한다. 먼저 복합단어는 대소문자, "/" 등과 같은 구분 문자를 사용하여 텀(term)들로 분리하고 스테밍(stemming)과 불용어(stop-word) 필터링을 수행한다. 이때 숫자로만 구성된 동적 런타임 값들은 검색에 큰 의미가 없으므로 미리 제거한다. 그 다음 단어 내 약어(abbreviation)들이 확장되고 동의어(synonym)를 발견하기 위해 WordNet과 같은 시소러스(thesaurus)를 사용한다. 이러한 과정을 통해 두 속성 A와 B 사이의 유사도은 다음과 같이 계산된다.

$$Sim(A,B) = \frac{2 \times \| Match(A,B) \|}{t_1 + t_2}$$

여기서, t_1과 t_2는 각 속성 안에 있는 텀들의 개수를 의미하고 $\| Match(A,B) \|$는 매칭 텀들의 개수를 리턴한다. 결과는 0과 1 사이의 실수 값을 가진다.

매개변수 유사도: 리소스 입출력 매개변수들 간의 유사도를 측정하는 것은 쉬운 일이 아니다. 왜냐하면, 리소스 내에는 매개변수들이 몇 개 존재하지 않으며, 하나의 리소스 내에 같은 매개변수는 거의 발생되지 않기 때문에 기존의 TF/IDF 방법[10]을 그대로 적용하기는 어렵다. 따라서 param name 값들에 대해 전처리 과정을 수행한 후 관련성이 많은 텀들을 개념으로 클러스터링하고 이런 텀들을 단어의 백으로 재배치한 후 TF/IDF 방법을 적용한다. 여기서 매개변수 클러스터링은 우리가 이미 제안하였던 RESTful 웹 서비스 온톨로지 학습 방법[11]을 이용한다. 하나의 질의와 저장소로부터 매치되는 임의의 후보 리소스 쌍을 (Q, R)이라 하고, 매개변수 Q와 R에는 각각 m과 n개의 매개변수들이 있다고 가정하자. 즉,

$$Q = q_1, q_2, \cdots q_i, \cdots q_m$$

$$R = r_1, r_2, \cdots r_j, \cdots r_n$$

Q의 q_i와 R의 r_j 간의 매치를 고려할 때, 클러스터링 기반 매개변수 유사도는 다음과 같이 매개변수 쌍들의 평균 값으로 계산된다.

$$Sim(Q,R) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} Matching(q_i, r_j)}{m+n}$$

여기서, $Matching(q_i, r_j) = \max\{TF/IDF(q_i, r_j)\}$ *for all* $1 \le i \le m,\ 1 \le j \le n$.

전체적으로 질의와 저장소에 있는 임의의 리소스 간의

유사도는 다음과 같이 계산된다.

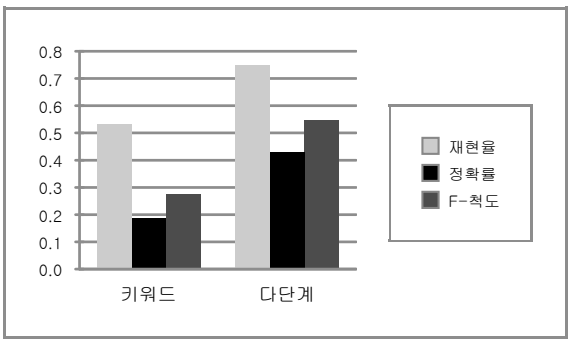
$$Similarity = \alpha(Sim_p) + \beta(Sim_i) + \gamma(Sim_o)$$

여기서, Sim_p는 path 속성, Sim_i는 입력, Sim_o는 출력 유사도이며, α, β, γ는 각각의 가중치이고 $\alpha+\beta+\gamma \leq 1$이다. 유사도 결과 값은 0과 1 사이의 실수 값을 리턴한다.

4. 실험 분석

실험 분석의 목적은 본 논문에서 제한하는 다단계 리소스 매칭 방법의 우수성을 보이는 것이다. 전통적인 RESTful 리소스 발견 방법은 키워드 기반 검색만 지원하고 있다. 이러한 키워드 검색 방법에 비해 다단계 리소스 매칭 방법이 얼마나 효율적으로 수행되는지 이들 두 방법을 비교/분석 하였다. 평가 방법은 정보 검색에서 가장 보편적으로 활용되고 있는 재현율, 정확률, 그리고 F-척도를 사용한다. 실험은 ProgrammableWeb 사이트로부터 다운로드 받은 168개의 RESTful 웹 서비스들에 대해 수행하였으며, 수집된 데이터는 264개의 request/response 엘리멘트로 구성되어 있다.

(그림 2)는 기존의 키워드 검색 방법과 비교하여 다단계 리소스 매칭 방법의 성능 향상을 보여주고 있다. 제안된 방법은 method 매칭, path 속성 매칭, 매개변수 유사도 측정으로 구성되어 있는데, 이 방법의 효과를 분석하기 위해 키워드 방법과 비교한 재현율, 정확률, 그리고 F-척도를 측정하였다. 실험 분석 결과 본 논문에서 제안한 다단계 리소스 매칭 방법이 기존의 키워드 검색 방법에 비해 재현율, 정확률, F-척도 각각 22%, 25%, 27% 개선된 것을 알 수 있다.



(그림 2) 다단계 리소스 매칭 방법의 성능 분석

5. 결론

본 논문에서는 RESTful 웹 서비스를 위한 WADL 기반 다단계 리소스 매칭 방법을 제안하였다. 본 연구의 핵심 내용은 같은 메소드를 포함하는 리소스들만 빠르고 쉽게 필터링 한 후 리소스 path 속성과 매개변수 유사도 매칭을 수행하였는데, path 속성 매칭은 구문 분석 방법을 사용하였고, 매개변수 유사도 매칭에서는 이미 우리가 제안하였던 매개변수 클러스터링 방법을 적용하였다. 제안된 방법은 실제 Open API 저장소인 ProgrammableWeb 사이트로부터 168개의 RESTful 웹 서비스를 다운로드 받아 실험 분석을 수행하였으며, 그 결과 기존의 키워드 검색 방법에 비해 재현율, 정확률, F-척도 각각 22%, 25%, 27%의 성능 향상을 보였다.

향후 후속 연구로는 RESTful 웹 서비스를 위한 리소스 조합 방법에 관한 연구를 수행할 필요가 있으며, 현재 구현 방법에서는 response의 representation 엘리먼트는 고려하지 않았는데, 이는 비록 이런 복잡한 문서정보를 고려하더라도 완전한 결과를 도출하는 데는 크게 도움을 주지 못하기 때문에 이를 제외시켰다.

참고문헌

[1] L. Richardson and S. Ruby, RESTful Web Services: Web services for the real world, O'Reilly Media, 2007

[2] R. Fielding, Architectural Styles and The Design of Network-based Software Architectures, PhD thesis, University of California, 2000

[3] http://www.programmableweb.com

[4] D. Martin, et al., "OWL-S: Semantic Markup for Web Services," http://www.w3.org/Submission/OWL-S/, 2004

[5] J. Bruijin, et al., "Web Service Modeling Ontology (WSMO)," http://www.w3.org/Submission/WSMO/, 2005

[6] 박유미, 문애경, 유현경, 정유철, 김상기, "SOAP 기반 웹 서비스와 RESTful 웹 서비스 기술 비교," 전자통신동향분석, 제25권 제2호, 2010년 4월

[7] M. Hadley, "Web Application Description Language (WADL)," http://www.w3.org/Submission/wadl/, 2009

[8] T. Steiner, "REST Describe & Compile," http://code.google.com/p/rest-api-code-gen/

[9] 이용주, "클러스터와 온톨로지 정보를 이용한 웹 서비스 매칭 알고리즘," 한국인터넷정보학회논문지, 제11권 제1호, 2010년 2월, pp. 59-69

[10] G. Salton and M. J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, 1983

[11] Y. J. Lee and C. S. Kim, "Building Semantic Ontologies for RESTful Web Services," 6th International Conference on Next Generation Web Services Practices (NWeSP2010), November 2010, pp. 37-40