디지털산업정보학회 논문지 제7권 제3호-2011년 9월

스마트 매쉬업을 위한 시맨틱 기반 Open API 온톨로지 구축 기법*

이 용 주**

Building Open API Ontologies based on Semantics for Smart Mashup

Lee, Yong Ju

〈Abstract〉

Recently, Open APIs are getting attention with the advent of Web 2.0. Open APIs are used to combine services and generate new services by Mashup. However, the growing number of available Open APIs raises a challenging issue how to locate the desired APIs. We automatically build ontologies from WSDL, WADL, HTML, and their underlying semantics. The key ingredient of our method is a technique that clusters input/output parameters in the collection of API methods into semantically meaningful concepts, and captures the hierarchical relationships between the terms contained in a parameter. These semantic ontologies allow search engines to support a similarity search for Open APIs based on various protocols such as SOAP, REST, JavaScript, and XML-RPC, and significantly improve the quality of APIs matching by the clustering and hierarchical relationships mechanism.

Key Words : Open API, Mashup, Ontology Learning, API Discovery, SOAP, REST

Ⅰ. 서론

최근에 다수의 Open API(Application Program Interface)들을 조합하여 만드는 매쉬업(Mashup)이 미래 IT 융합 서비스의 효과적인 구현 방법으로써 그 관심도가 점점 높아지고 있다. 매쉬업을 제작하기 위해서는 그 목적에 맞는 적합한 Open API들을 먼저 찾아야 하는데, 이러한 작업은 인터넷 상에 Open API들의 수가 기하급수적으로 증가됨에 따라 그렇게 쉬운 작업이 아니다. 현재 대부분의 Open API 포털 사이트(예, programmableWeb[1], SeekDa[2])들은 카테고리별 검색 또는 키워드 검색만 제공하고 있다.

카테고리별 검색 방법은 분류 코드에 속하는 서비스들을 빠르고 쉽게 필터링하여 탐색 범위를 한정시켜 준다. 그러나 이 방법은 검색 결과가 너무 광범위하여 원하는 결과를 쉽게 찾을 수가 없다. 예를 들어, programmableWeb 사이트에서 'mapping'에 관한 카테고리별 검색 결과는 206개의 API가 나타났다. 또한, 키워드 검색 방법도 이미 여러 연구에서 밝혀진 것과 같이 나쁜 재현율과 나쁜 정확률 때문에 문제가 많다. 즉, 키워드 검색에서는 키워드와 정확히 일치하는 API들만 발견되므로 원하는 결과가 누락되거나 전혀 관련 없는 결과들이 불필요하게 포함될 수 있다.

* 이 논문은 2010학년도 경북대학교 전임교원 연구년 교수 연구비에 의하여 연구되었음

** 경북대학교 컴퓨터정보학부 교수(교신저자)

이러한 기존의 검색 방법들의 한계를 극복하기 위한 기법으로서 시맨틱 기반 온톨로지(ontology) 활용 방법이 있을 수 있다(예, OWL-S[3], SAWSDL[4], SA-REST[5] 등). 웹 서비스 저장소에 추가적인 시맨틱 정보를 주석처리(annotation)하여 키워드에 일치하지 않은 API 일지라도 의미적으로 연관성 있는 API들을 확장·탐색할 수 있다. 그렇지만 이러한 온톨로지는 대부분 전문가들의 수작업으로 구축되어야 하므로 시간 및 인적 제약 때문에 실질적인 온톨로지를 구축하기가 어렵다. 또한 현시점에서 수많은 Open API 전체에 대해 주석을 다시 단다는 것은 거의 불가능하게 보인다.

본 논문에서는 Open API를 개발할 때 자동으로 생성되는 웹 서비스 기술언어(예, WSDL[6], WADL[7])만 가지고 항목 간 숨어있는 시맨틱 정보를 찾아내어 관련 온톨로지를 자동 구축하는 하나의 새로운 시맨틱 기반 Open API 온톨로지 구축 기법을 제안한다. 이를 통해 시맨틱 정보를 위한 주석처리 부담을 줄일 수 있고, 보다 효율적인 API 검색을 지원한다. 즉, 키워드가 정확하게 일치하지 않더라도 사용자가 원하는 API들을 검색할 수 있고, 반대로 키워드가 일치하더라도 사용자가 원하지 않은 API들은 검색 결과에서 제거된다.

기존 온톨로지 구축 방법과 다른 특이한 사항은 저장소 내에 있는 Open API들이 일반적으로 하나가 아닌 다양한 프로토콜을 채택하고 있는 점이다. Open API와 매쉬업에 대한 대표적인 웹 사이트인 programmableWeb에 의하면, 2011년 7월 25일 기준으로 3,517개의 Open API가 존재하고 있는데, 이들은 <그림 1>과 같이 REST, SOAP, JavaScript, XML-RPC 등 다양한 프로토콜로 제공되고 있다. 현재 REST 프로토콜로 제공되는 서비스 수가 72%로 가장 많지만, 보안을 중시하는 공공기관이나 기업 내부의 서비스는 점점 SOAP 프로토콜로 단일화되어 가고 있는 추세이며, 가장 많이 활용되고 있는 mapping 관련 Open API는 주로 JavaScript로 제공되고 있다. 따라서 제안되는 기법은 REST 뿐만 아니라 SOAP을 비롯한 다양한 프로토콜을 대상으로 통합 검색이 가능하도록 해야 한다.

<그림 1> 다양한 Open API 프로토콜

본 논문의 구성은 다음과 같다. 2장에서 Open API 프로토콜을 간단히 살펴보고 3장에서 시맨틱 기반 Open API 온톨로지 구축 기법을 제안한다. 4장에서 유사 API 발견 기법을 기술하고 5장에서 관련 연구를 설명한다. 그리고 6장에서 결론을 내린다.

Ⅱ. Open API 프로토콜

Open API는 일반적으로 REST, SOAP, JavaScript, XML-RPC 등 다양한 프로토콜을 통해서 제공된다. 이러한 프로토콜에 따라 메시지 전송 방식과 서비스 제공 형식이 다를 수 있기 때문에 각 프로토콜별 웹 서비스 특성을 조사할 필요가 있다.

2.1 SOAP 기반 웹 서비스

SOAP 기반 웹 서비스는 표준적인 웹 프로토콜을 통해 웹 환경 분산 컴퓨팅을 가능케 하며, CORBA나 DCOM과 같은 전통적인 분산 컴퓨팅 모델을 대체할 수 있는 새로운 기술로써 현재 많은 관심을 받고 있다. SOAP 기반 웹 서비스 기술은 SOAP, WSDL, 그리고 UDDI로 구성되어 있는데, SOAP은 분산 컴퓨팅 환경에서 컴포넌트 간의 정보를 교환하기 위해 사용되는 XML 기반의 표준 프로토콜이며, WSDL은 웹 서비스 이용에

필요한 인터페이스와 입/출력 메시지 형식 등을 기술하고 있으며, UDDI는 웹 서비스를 등록하고 검색하기 위한 저장소(registry)이다[8]. UDDI는 키워드에 의한 사전 분류시스템이기 때문에 웹 서비스 조합 및 시맨틱 기능은 제공하지 않는다. 이러한 기능을 제공하기 위해서는 기존의 표준 웹 서비스 기술에 새로운 레이어가 추가되어야 한다.

WS-BPEL은 웹 서비스 조합을 위해 제안된 스팩으로써 기존의 WSFL과 XLang 조합 언어의 장점을 취합한 비즈니스 프로세스 정의 언어이다. 비즈니스 프로세스는 비즈니스 목표를 실현하기 위해 액티비티(activity)들의 그룹으로 표현될 수 있으며, 이러한 액티비티들은 제어 구성자(control construct)를 사용하여 좀 더 복잡한 프로세스로 조합될 수 있다. WS-BPEL은 IBM, 마이크로소프트, BEA 등 대형 IT 업체를 중심으로 활발하게 연구되고 있으나 시맨틱 개념이 부족하여 웹 서비스 조합이 사전에 수동으로 수행되어야 하는 단점이 있다.

시맨틱 웹을 실현하기 위한 온톨로지는 웹 상에 존재하는 자원들에 대한 지식 표현 방법으로서 W3C 표준인 RDF를 기반으로 온톨로지 언어를 통해 기술된다. 잘 알려진 웹 온톨로지 언어로는 DAML, OIL, SHOE 등이 있으며 DAML과 OIL의 기능을 합쳐서 만들어진 DAML+OIL이 있다. 현재는 이를 발전시킨 OWL에 대한 표준화가 진행되고 있으며, OWL-S는 OWL을 기반으로 지능적인 웹 서비스가 가능하도록 한 시맨틱 언어이다.

SOAP 기반 웹 서비스는 느슨하게 연결된(loosely coupled) 컴포넌트들의 집합을 통하여 서비스를 구성하기 때문에 서비스 조합 생성 시 뛰어난 재사용성과 확장성을 보여준다. 또한 W3C 표준에 의거한 SOAP, WSDL, 그리고 WS-* 와 같은 표준을 활용함으로써 보다 비즈니스 프로세스에 집중할 수 있도록 지원한다. 그러나 표준의 범위가 계속해서 확대됨에 따라 점차 복잡해지게 되었으며, 이에 따라 너무 많은 표준에 의해 제약 사항이 증가하였고 개발의 복잡성도 증가하는 문제점을 초래하였다.

2.2 RESTful 웹 서비스

SOAP 기반 웹 서비스의 구축비용에 대한 회의적인 시각이 확산됨에 따라 이를 보완할 수 있는 RESTful 웹 서비스에 대한 관심이 증가하였다. RESTful 웹 서비스의 최초 시작은 SOAP 기반 웹 서비스의 복잡함에 얽매이지 않고 쉬운 방식으로 웹 서비스를 구축하기 위한 REST 아키텍처의 연구와 함께 시작되었다.

REST는 웹의 창시자 중 한 사람인 Roy Fielding의 박사학위 논문[9]에 의해 소개되었다. 그는 현재의 웹 아키텍처가 웹이 지닌 본래의 설계 우수성을 충분히 활용하지 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(resource)의 URI 설정: RESTful 웹 서비스의 가장 큰 특징 중의 하나는 모든 대상을 리소스, 즉, 자원으로 표현한다는 것이다. 이 리소스는 HTTP URI(Uniform Resource Identifier)에 의해 표현되며, 웹 사이트, 블로그, 이미지, 음악, 이용자, 지도, 검색결과 등 웹에서 다른 이들과 공유하고자 개방된 모든 자원을 의미한다.
- HTTP 메소드 사용: REST 구조에서의 리소스는 HTTP의 기본 메소드인 POST, GET, PUT, DELETE 만으로 접근할 수 있다. 리소스에 접근하기 위한 이러한 4개의 HTTP 메소드는 일반 CRUD(Create, Read, Update, Delete) 오퍼레이션에 각각 대응될 수 있다.
- 다양한 표현(representation) 방식: HTTP의 기본 메소드로 전달되는 리소스는 다양한 방식으로 표현되는데, 이는 XML, JSON, HTML, 텍스트, 이미지 등이 가능하며 클라이언트에서 원하는 형식으로 표현하면 서버에서 이를 처리하게 된다. 리소스의 다양한

표현 방식은 HTTP의 accept 헤더값 또는 URI 파라미터로 지정할 수 있다.

- 스테이트리스(stateless): HTTP의 특성을 상속하여 RESTful 웹 서비스 역시 스테이트리스 특성을 가지게 되는데, 스테이트리스란 웹 서비스 제공 서버 측에서 클라이언트의 상태(state) 정보를 저장, 관리하지 않는 것을 의미한다. 즉, 모든 HTTP 요청은 완벽한 고립 상태에서 발생한다. 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 Open API들이 RESTful 웹 서비스로 작성되어 있으며, 이는 위젯(widget)을 이용한 매쉬업을 활성화시킨 원동력이 되었다[10].

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 WADL을 발표하게 되었고, 간단하다는 장점 때문에 개발자들 사이에서 WADL의 사용은 점차 증가되고 있는 실정이다.

2.3 JavaScript와 XML-RPC

JavaScript는 Netscape사와 Sun사가 공동으로 개발한 HTML과 함께 사용할 수 있는 스크립트 언어로 운영체제에 독립적으로 실행될 수 있다. JavaScript는 작고 쉬운 스크립트 언어로서 HTML 문서 안에 포함되어서 컨트롤러, 이벤트 등을 이용해 동적인 웹 페이지를 만들 수 있도록 해준다. 우리가 보통 JavaScript를 말할 때는 클라이언트 및 서버측 스크립트가 코드 내에 병합된 형태를 말한다.

RPC는 Remote Procedure Call의 약자로 분산 컴퓨터 환경에서 이기종 컴퓨터 자원을 사용하는 기술이다. RPC는 원격의 컴퓨터도 마치 자신의 컴퓨터에 존재하는 함수를 호출하는 것처럼 프로그램할 수 있게 만들어 준다. XML-RPC는 이 개념을 웹으로 옮겨온 것으로 프로그램에서 함수 호출을 하듯이 원격에 있는 사이트에 정보를 요청하고 받아올 수 있게 해 준다. 이때 주고받는 인자와 리턴 값은 XML로 인코딩되고, 실제로 데이터를 전송하는 수송 수단으로는 범용적인 HTTP(POST)를 사용한다. 국내에서는 다음(Daum)이 자사의 블로그 API를 XML-RPC 형태로 제공하고 있다.

JavaScript와 XML-RPC에 관한 기술 언어는 현재 제안된 바가 없으며, 다만, 서비스에 대한 설명과 사용 방법 등을 기술한 HTML 페이지를 주로 제공하고 있다.

Ⅲ. Open API 온톨로지 구축 기법

3.1 시스템 구조

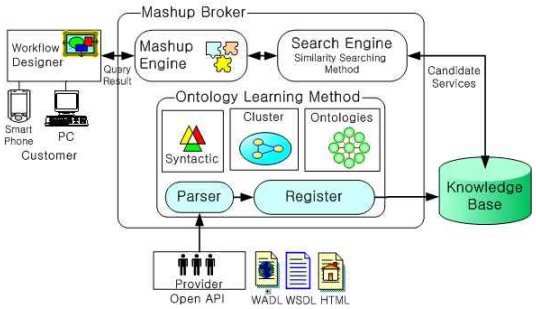
본 논문에서 제안하는 Open API 온톨로지 구축 기법은 이전에 이미 우리가 제안하였던 SOAP 기반 온톨로지 학습 방법[11]과 RESTful 온톨로지 학습 방법[12]에 기반하여 수행된다. 이들 두 방법의 큰 차이점은 웹 서비스 기술언어인 WSDL과 WADL의 사용이다. SOAP 기반 웹

서비스에서는 웹 서비스를 개발할 때 WSDL 문서가 자동 생성되는 반면에, RESTful 웹 서비스에서는 REST 본래의 목적이 단순함이었기 때문에 애초에 WSDL과 같은 기술 언어를 요구하지 않는다. 본 논문에서는 RESTful 웹 서비스의 기술 언어로써 WADL을 채택하였으나, RESTful 온톨로지를 구축하기 위해 WADL이 꼭 필요한 것은 아니다. 단지 기계 판독 가능한(machine readable) WADL 형태의 문서는 자동화에 도움을 줄 수 있는 수단이 될 수 있다. WADL 문서가 없는 경우에는 일반적으로 RESTful 웹 서비스 사용 방법을 제공하고 있는 HTML 설명 페이지로부터 온톨로지 구축에 필요한 정보를 수작업으로 추출한다. 이는 JavaScript와 XML-RPC 서비스에 대해서도 동일한 방법으로 수행될 수 있다.

SOAP 기반 웹 서비스는 기계 판독 가능한 WSDL 형태로 서비스 정보를 제공하기 때문에 이를 파싱(parsing)하여 바로 온톨로지 학습 방법에 적용할 수 있다. 그러나 RESTful 웹 서비스와 기타 프로토콜(즉, JavaScript, XML-RPC)로 제공되는 서비스는 설명 정보들이 각각의 웹 사이트를 통해 사람 판독 가능한(human readable) HTML 형태로 제공되고 있다. 따라서 수작업으로 직접 서비스 정보를 취득하여 기계 판독 가능한 형태로 초기 데이터셋을 구축한 다음 온톨로지 학습 방법을 적용한다. 하지만 RESTful 웹 서비스가 WADL 형태로 초기 데이터셋을 제공한다면 SOAP처럼 바로 온톨로지 학습 방법을 적용할 수 있다. 1)

<그림 2>는 전체적인 스마트 매쉬업 시스템의 구조를 설명하고 있다. 본 시스템은 서비스 공급자(provider)와 사용자(customer), 그리고 매쉬업 중계자(Mashup broker)로 구성되어 있다. 매쉬업 중계자는 매쉬업 엔진(Mashup engine), 탐색 엔진(search engine), 온톨로지 학습 방법(ontology learning method) 모듈로 이루어져 있으며, 중계자는 전체 시스템의 미들웨어(middleware) 역할을 수행한다. 스마트 매쉬업 시스템은 현재 개발 중에 있으며 본 논문에서는 이들 중 온톨로지 학습 방법과 탐색 엔진 부분에 대해 중점적으로 기술한다.

1) REST 프레임워크인 Restlet 2.0[13]에서는 WADL 기능을 제공한다.



<그림 2> 스마트 매쉬업 시스템의 구조

3.2 온톨로지 학습 방법

일반적으로 Open API 제공자는 포털 사이트 내에 자신의 웹 서비스와 관련된 WSDL/WADL/HTML 문서, 그리고 그 서비스에 대한 간단한 설명을 등록함으로 API를 공개한다. 각 문서들을 살펴보면, WSDL에서 하나의 웹 서비스는 오퍼레이션들의 집합으로 구성되어 있고, 각 오퍼레이션은 다수의 입/출력 매개변수(parameter)들로 이루어져 있다. WADL에서는 하나의 웹 서비스가 리소스들의 집합으로 구성되어 있고, 각 리소스는 다수의 request/response 매개변수들로 이루어져 있다. 그리고 HTML 문서에서는 주로 Open API의 사용법과 입출력 매개변수들에 대한 설명을 포함하고 있다. 따라서 Open API 포털 사이트로부터 다음과 같은 공통의 정보를 얻을 수 있다. 참고로 본 논문에서는 편의상 오퍼레이션과 리소스는 메소드로, 입/출력과 request/response는 입출력으로 표기한다.

- Open API 이름과 이에 대한 간단한 텍스트 설명
- API 메소드 이름과 이에 대한 텍스트 설명
- 매개변수 정보(메소드의 입출력은 각각 매개변수들의 집합으로 구성되어 있음)

위와 같은 공통의 정보로부터 온톨로지 구축에 필요한 자료를 대부분 취득할 수 있다. 예를 들면, OWL-S는 가장 최근에 발표된 시맨틱 기반 웹 서비스 언어인데, OWL-S 온톨로지는 기계 가독형으로 웹 서비스 기능을 묘사할 수 있는 메카니즘을 제공한다. 이러한 메카니즘은 자동화된 API 발견 및 매쉬업을 가능하게 만든다. OWL-S는 서비스를 기술하기 위해 서비스 프로파일, 서비스 모델, 그리고 서비스 그라운딩 세가지 클래스로 구성되어 있으며, 서비스 프로파일은 에이전트가 서비스를 탐색하는데 필요한 정보를 제공하고 있으며, 서비스 모델 및 그라운딩은 에이전트가 서비스를 사용할 수 있도록 충분한 정보를 제공하고 있다[3]. 그러므로 Open API 탐색 및 매쉬업 문제를 해결하기 위해서는 주로 서비스 프로파일을 사용하게 된다.

서비스 프로파일은 서비스 제공자에 대한 정보(조직의 연락처 및 관련된 정보), 서비스를 처리하기 위한 기능에 대한 정보(서비스 입출력 정보), 그리고 서비스의 특성을 구체화한 특징에 관한 정보(서비스 카테고리 및 품질등급 정보)를 포함하고 있다. 따라서 이러한 정보들은 위에서 설명한 Open API 포털 사이트에 등록된 WSDL/WADL/HTML 문서로부터 OWL-S 서비스 프로파일 내용을 취득할 수 있다. 다만, 이들 문서에서는 구문(syntactic) 정보만 있을 뿐 온톨로지에서 요구되는 동일 또는 계층 관계와 같은 객체지향 클래스에 대한 정보(예, Customer **equivalent** Person 또는 Sedan **subclassOf** Car)가 없다. 따라서 본 연구는 WSDL/WADL/HTML 구문 정보만 가지고 항목 간의 숨어 있는 시맨틱 정보를 찾아내어 온톨로지를 자동 구축하고 이를 이용한 유사 API들을 발견하는 것이다.

특히, 본 연구에서 제안하는 온톨로지 학습 방법은 완전히 다른 접근 방법으로 구현되는 다양한 프로토콜 기반 Open API들에 대해 동일한 절차로 시맨틱 개념을 추가할 수 있는 통합된 온톨로지 구축 방법을 제공한다. 지금까지 온톨로지에 관한 연구는 거의 대부분 SOAP 기반이었고 본 논문과 같은 통합된 접근 방법은 거의 찾아볼 수 없다. <표 1>은 각 프로토콜별 제안되는 기술 언어 레이어를 비교한 것이다. 중요한 사항은 표준 언어만 가지고 추가 부담 없이 시맨틱 정보를 자동 구축할 수 있는 것이다.

<표 1> 프로토콜별 기술 언어 레이어

프로토콜	엑세스	구문정보	시맨틱
SOAP	SOAP	WSDL	제안된 온톨로지 학습방법
REST	HTTP	WADL	
기타	HTTP	HTML	

다음 절부터는 기존에 우리가 제안하였던 SOAP 기반 온톨로지 학습 방법[11]을 간단히 살펴보고 다른 프로토콜들을 지원하기 위한 확장된 부분을 자세히 설명한다.

3.2.1 매개변수 클러스터링

매개변수들을 토큰화하여 용어들로 분리한 후, 관련성이 많은 용어들에 대해 클러스터를 형성하면 이 클러스터는 각각의 단어가 아닌 하나의 의미 있는 개념(concept)을 나타낸다. 이러한 클러스터는 "매개변수들이 동시에 자주 나타난다면, 그것들은 같은 개념을 나타내는 경향이 있다"는 가정 하에 하나의 특별한 연관규칙(association rules)[14-15]에 따라 만들어 진다.

연관규칙은 용어 A가 일어나면 용어 B가 일어난다는 의미로 A ⇒ B로 표현될 수 있으며, 여기서 트랜잭션(transaction)은 웹 서비스 입출력에 나타나는 용어들의 집합으로 볼 수 있다. 지지도(support)와 신뢰도(confidence)는 해당 규칙이 얼마나 유용한지를 나타내는 지표로서, 지지도는 용어 A와 B를 동시에 포함하는 트랜잭션의 확률을 표현하며, 신뢰도는 A가 주어졌을 때 B가 동시에 나타날 트랜잭션의 확률을 나타낸다.

$$support(A \Rightarrow B) = P(A \cup B)$$

$$confidence(A \Rightarrow B) = \frac{P(A \cup B)}{P(A)}$$

연관규칙을 찾는 과정은 기본적으로 빈발 용어 집합(frequent termset)을 찾는 단계와 연관규칙을 생성하는 두 단계로 구성된다. 빈발 용어 집합이란 후보 용어 집합(candidate termset) 중 최소 지지도(minimum support: *minsupp*) 이상의 값을 가진 용어 집합으로서 데이터 마이닝 기법에서 생성되는 연관규칙의 단위가 된다. 빈발 용어들이 생성되고 나면 이들로부터 최소 신뢰도(minimum confidence: *minconf*)를 만족하는 모든 연관규칙들을 찾는다. 이러한 연관규칙 탐사 문제는 Apriori[16] 알고리즘을 사용하면 효율적으로 처리될 수 있다.

클러스터링 과정에서 가장 이상적인 클러스터를 형성하기 위하여 다음의 두 가지 클러스터 특성을 고려한다. 즉, 클러스터의 cohesion(한 클러스터 내의 용어들과의 응집력)은 높고, 클러스터 간의 correlation(다른 클러스터 용어들 간의 상호관계)은 낮아야만 한다. 클러스터 C_1이 주어졌을 때 cohesion(C_1)은 한 클러스터 내에 서로 밀접하게 연관되어 있는 용어 쌍들의 분포 확률로 정의된다.

$$cohesion(C_1) = \frac{\| associRule(i \Rightarrow j) \|}{\| C_1 \| \| (C_1 - 1) \|}$$

여기서, $i, j \in C_1, i \neq j$, $associRule(i \Rightarrow j) = \{support(i \Rightarrow j) > minsupp \ \& \ confidence(i \Rightarrow j) > minconf\}$. 또한, 클러스터 C_1과 C_2가 주어졌을 때 correlation(C_1, C_2)은 클러스터 간 서로 밀접하게 연관되어 있는 용어 쌍들의 분포 확률로 정의된다.

$$correlation(C_1, C_2) = \frac{\| associRule(i \Rightarrow j) \| + \| associRule(j \Rightarrow i) \|}{2 \| C_1 \| \| C_2 \|}$$

여기서, $i \in C_1, j \in C_2$

마지막으로 전체적인 클러스터 C의 품질을 측정하기 위한 점수(score)는 다음과 같이 정의된다.

$$score(C) = \frac{avg(cohesion)}{avg(correlation)} = \frac{(\| C \| - 1) \sum_{C_1 \in C} cohesion(C_1)}{2 \sum_{C_1, C_2 \in C} correlation(C_1, C_2)}$$

이러한 과정에서 우리의 최종 목표는 가장 높은 점수를 갖도록 클러스터를 형성하는 것이다.

한편, 이러한 SOAP 기반 온톨로지 학습 방법은 REST나 JavaScript/XML-RPC를 위해서는 이들이 잘 맞지 않는다. 왜냐하면, SOAP 기반 웹 서비스는 WSDL을 매개체로 오퍼레이션 능력을 제공하고자 한 기술이고, RESTful 웹 서비스는 URI를 기반으로 리소스를 개방하고자 하는 기술이다. JavaScript/XML-RPC는 해당 클래스를 호출하는 방식으로 자바와 같은 프로그래밍 개념과 동일하다. JavaScript/XML-RPC는 본 연구에서는 수작업으로 매개변수 클러스터링에 필요한 트랜잭션 정보를 추출하기 때문에 이후부터는 SOAP 기반 웹 서비스와 동일하게 취급한다.

본 기법과 관련하여 SOAP과 RESTful 웹 서비스 간의 가장 다른 점은 RESTful 웹 서비스의 리소스는 XML, JSON, HTML 등 다양한 방식으로 표현되는 것이다. WADL에서 오퍼레이션은 resource 엘리멘트로 대응되고, 입/출력 매개변수는 request/response의 param 엘리멘트와 매치된다. 그러나 response 엘리멘트는 representation 엘리멘트가 있을 수 있는데, 이는 매쉬업 데이터 디스플레이에 사용되고 매개변수 클러스터링 기법에서는 사용될 수 없으므로 트랜잭션 집합에서 제외한다.

3.2.2 매개변수 패턴 분석

본 기법의 주된 아이디어는 매개변수 내의 각 단어들

사이의 상관관계를 취득하고, 비교되는 단어들이 서로 유사하고 상관관계가 조건에 일치한다면 그 비교를 매치하는 것이다. 이러한 구축 기법은 "사람들이 단어를 조합하여 복합단어로 된 매개변수를 만들 때 일반적으로 비슷한 패턴을 사용한다[17-18]"는 관찰로부터 시작한다. 일반적으로 사람들이 어떤 한 개념을 표현할 때 다양한 방법들이 있을 수 있지만 공간적인 제약 하에서는 비슷한 패턴을 사용하는 경향이 있다.

하지만, 이러한 패턴들은 사용되는 프로토콜이나 도메인에 따라 다를 수 있는데, 이를 조사하기 위해 본 논문에서는 SOAP 기반 웹 서비스와 RESTful 웹 서비스에 대해 각각 별도의 실험 데이터를 만들어서 분석해 보았다. 먼저 SOAP 기반 웹 서비스는 SOAP 관련 웹 서비스 저장소인 xmethods.net[19]에서 WSDL 파일을 다운로드 받아 사전 작업 처리과정을 거쳐 패턴 분석을 수행할 수 있는 데이터 파일을 준비하였다. xmethods.net에서는 약 600여개의 WSDL 파일이 존재하고 있으나 본 조사에서는 모든 파일을 이용하지 않고 zip, weather, address 도메인에 관한 50개의 파일에 대해서만 실험을 수행하였다. 다음으로 RESTful 웹 서비스에 대해서는 OpenAPI 및 매쉬업 저장소인 ProgrammableWeb[1]으로부터 RESTful 웹 서비스 매개변수들을 다운로드 받아 실험 분석을 수행하였다. 이 사이트는 본 논문의 실험 시점에 3,517개의 Open API가 존재하고 있었으며, 이들 중 REST 방식으로 구현된 웹 서비스는 2,518개였다. 본 조사에서는 이들 모든 웹 서비스들을 다 이용하지 않고 mapping, travel, weather 도메인에 있는 168개의 RESTful 웹 서비스에 대해서만 실험을 수행하였다. 이런 샘플링 개수는 <그림 1>의 Open API 프로토콜별 서비스 개수 비율에 어느 정도 맞춘 것이다.

이들 실험 데이터에 POS(part-of-speech) 형태소 분석기[2]를 적용시킨 결과 <표 2>와 같은 형태를 취하였다. 분석 결과 SOAP 기반 웹 서비스에서는 단지 하나의 토큰으로 구성된 매개변수(예, City)가 38%로 가장 많았고, 명사+명사(37%), 명사+명사+명사(14%), 동사+명사(6%), 명사+전치사+명사(3%), 형용사+명사(1%), 그리고 기타(1%) 순으로 나타났다. RESTful 웹 서비스에서는 단지 하나의 토큰으로 구성된 매개변수는 전체의 44%를 차지하였으며, 명사+명사(30%), 형용사+명사(9%), 동사+명사(7%), 명사+명사+명사(6%), 명사+전치사+명사(5%), 그리고 기타(0.3%) 순으로 나타났다. 본 결과로부터 특이한 사항은 다른 프로토콜을 사용하거나, 다른 도메인을 선택하더라도 매개변수 패턴은 단지 출현 빈도의 순위에는 다소간의 변동이 있지만 도출된 5개의 패턴 종류는 동일한 사실을 알 수 있다.

<표 2> 복합단어 매개변수의 패턴

구분	패턴	출현수(%)	예
SOAP	Noun1+Noun2	470 (37%)	CompanyID
	Noun1+Noun2+Noun3	175 (14%)	TelephoneAreaCode
	Verb+Noun	70 (6%)	enrollAccount
	Noun1+Preposition+Noun2	40 (3%)	PasswordOfAccount
	Adjective+Noun	15 (1%)	virtualAccount
REST	Noun1+Noun2	2435 (30%)	CountryCode
	Adjective+Noun	752 (9%)	highTemperature
	Verb+Noun	608 (7%)	updateList
	Noun1+Noun2+Noun3	472 (6%)	TelephoneAreaCode
	Noun1+Preposition+Noun2	368 (5%)	PasswordOfAccount

따라서 <표 2>의 관찰로부터 매개변수에 대한 온톨로지 변환 규칙은 아래와 같이 5개의 규칙을 정할 수 있다. 규칙을 정하는 과정에서, 온톨로지 개념은 일반적으로 속성(property)과 상-하위(subclass) 관계로 표현되므로 [21] 본 연구에서는 각 단어들 간의 속성 및 상-하위 관계만 중점적으로 취급하였고, 규칙 순서는 중요하지 않으나 편의상 출현수가 많은 REST 순위에 따랐다.

규칙-1: 매개변수가 Noun1+Noun2일 경우

"Parameter **propertyOf** Noun1" 관계가 된다.

규칙-2: 매개변수가 Adjective+Noun일 경우

2) 본 논문에서는 CRFTagger[20] 형태소 분석기를 사용하였다.

"Parameter **subClassOf** Noun" 관계가 된다.

규칙-3: 매개변수가 Verb+Noun일 경우

"Parameter **subClassOf** Noun" 관계가 된다.

규칙-4: 매개변수가 Noun1+Noun2+Noun3일 경우

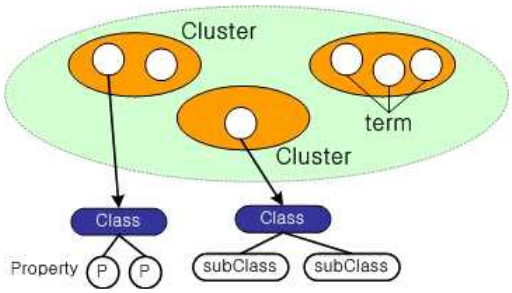
"Parameter **propertyOf** Noun1" 관계가 된다.

규칙-5: 매개변수가 Noun1+Preposition+Noun2일 경우

"Parameter **propertyOf** Noun2" 관계가 된다.

위와 같은 규칙을 사용하여 온톨로지가 구축되고 나면, 다음 단계는 질의문에 의해 두 개념 간 매칭을 시키는 것이다. 즉, 두 개의 온톨로지 개념이 ① 어떤 개념이 다른 개념의 속성일 경우(propertyOf), ② 어떤 개념이 다른 개념의 자식관계인 경우(subClassOf) 서로 매치된다.

참고로, 매개변수 클러스터링과 매개변수 패턴 분석 기법으로부터 도출된 결과를 도식으로 표현하면 <그림 3>과 같다. 먼저 클러스터를 예를 들어 설명하면, ZipCode와 관련된 클러스터는 {zip, city, area, state}와 {code, zip}과 같이 된다. 계층 구조는 Zip 용어 하위에 "ZipCode **propertyOf** Zip" 관계로부터 ZipCode가 이의 속성이 된다. 클러스터링 기법은 연관성이 높은 용어들을 한 클러스터에 묶어 동일한 개념처럼 취급하여 검색의 재현율을 향상시키며, 패턴 분석 기법은 ZipCode에서 Code만 포함하고 있는 서비스들을 배제시킴으로써 검색의 정확률을 향상시킨다.

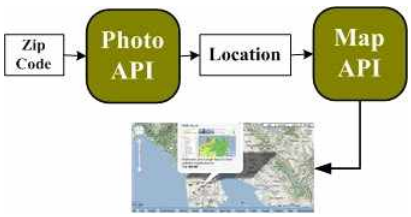


<그림 3> 클러스터와 계층구조 온톨로지 구조

Ⅳ. 유사 API 발견 방법

웹 상에 제공되는 Open API 수가 증가할수록 매쉬업에 필요한 API들을 선택하고 조합하는 일은 더욱 더 어려워 보인다. 따라서 본 장에서는 유사도 측정 방법을 통해 매쉬업에 필요한 API를 사용자가 원하는 순서대로 정렬된 결과를 제공하는 좀 더 지능적인 발견 방법을 제안한다.

매쉬업은 기본적으로 Open API의 입력과 출력 데이터를 연결하여 개발된다. 따라서 매쉬업을 구성하는 API 사이에는 반드시 연결의 매체가 되는 연관 관계가 있어야 한다. 예를 들면, <그림 4>와 같이 사진 서비스를 제공하는 Photo API를 지도 서비스를 제공하는 Map API와 매쉬업하기 위해서는 사진을 촬영한 위치와 같이 지도와 연결될 수 있는 공통의 매개체가 존재해야 한다. 그러나 두 API 사이에 사용되는 매개변수 어휘는 다양할 수 있다. 즉, Country, State, City, Area 등, 또한 City도 CityName, NameOfCity, CityCode 등 매우 다를 수 있다. 따라서 시맨틱 온톨로지를 활용하여 같은 개념의 매개변수들을 자동으로 매치시키고 유사도 측정을 통해 순위화된 유사 API들을 발견할 필요가 있다.



<그림 4> 사진과 지도 서비스를 결합한 매쉬업

다양한 프로토콜을 사용하는 Open API들에 대해 통합된 검색 방법을 제공하기 위해서는 공통의 질의 템플릿을 설정해야 한다. 이러한 템플릿은 3.2절에서 설명된 공통의 정보로부터 메소드 텍스트 설명과 입출력 매개변수로 구성하였다. 즉, 질의 템플릿 Q는 다음과 같이 3개의 튜플(tuple)로 표현하였다.

$Q = < desc, param(i), param(o) >$

여기서, *desc*은 메소드 텍스트 설명, *param(i)*는 입력 매개변수, *param(o)*는 출력 매개변수를 표시한다.

유사도 측정 방법은 참고문헌 [11]에서 제안한 유사 오퍼레이션 발견 방법을 이용하였으며, 여기서는 Open API에 대한 수정된 부분만 간단히 설명하고 자세한 부분은 참고문헌을 참조하기 바란다.

먼저, 메소드 텍스트 설명에 대한 유사도(Sim)를 측정하기 위해 정보검색 분야에서 널리 사용되고 있는 TF/IDF 방법[22]을 사용한다. 즉,

$$Sim(Q,D) = \frac{\sum_{k=1}^{t} w_{qk} \times w_{dk}}{\sqrt{\sum_{k=1}^{t} (w_{qk})^2 \times \sum_{k=1}^{t} (w_{dk})^2}}$$

여기서, 각 단어의 가중치(term weight) w_{dk}와 w_{qk}는 해당 문서에서 각 단어의 빈도(TF)와 역문헌빈도(IDF)의 곱으로 나타낸다.

다음으로, 입출력 매개변수에 대해서는 3.2절에서 서술된 바와 같이 클러스터링과 패턴 분석 기법이 적용된다. 패턴 분석 기법은 정제 단계에서 활용되므로 따로 설명하고 여기서는 단지 클러스터링 기법만 고려한다. 하나의 질의와 저장소로부터 매치되는 임의의 후보 메소드 쌍을 (Q, R)이라 하고, 매개변수 Q와 R에는 각각 m과 n개의 매개변수들이 있다고 가정하자.

$$Q = q_1, q_2, \cdots q_i, \cdots q_m$$
$$R = r_1, r_2, \cdots r_j, \cdots r_n$$

Q의 q_i와 R의 r_j 간의 매치를 고려할 때, 클러스터링 기반 매개변수 유사도(Similar)는 다음과 같이 매개변수 쌍들의 평균값으로 계산된다.

$$Similar = \frac{2 \times \sum_{i=1}^{m} PairSim(q_i, r_j)}{m+n}$$

여기서, $PairSim(q_i, r_j) = \max\{Sim(q_i, r_j)\}$ *for all* $1 \le j \le n$, $i = 1, 2, \cdots m$ 이다.

위 식을 이용하여 입력과 출력 매개변수 유사도를 각각 계산할 수 있으며, 전체적으로 질의 템플릿 Q와 저장소에 있는 임의의 메소드 R간의 유사도(Similarity)는 다음과 같이 계산된다.

$$Similarity = \frac{\alpha(DesSimilar) + \beta(InSimilar) + \gamma(OutSimilar)}{\alpha + \beta + \gamma}$$

여기서, *DesSimilar*는 설명 유사도, *InSimilar*는 입력 매개변수 유사도, 그리고 *OutSimilar*는 출력 매개변수 유사도이며, α, β, γ는 각각의 가중치이다. 결과 값은 0과 1 사이의 실수값을 리턴한다.

한편, 위의 유사도 측정 방법에서는 매개변수 패턴 분석 기법은 고려하지 않았으나, 이를 통한 온톨로지를 활용함으로써 검색 결과의 정확률을 향상시킬 수 있다. 즉, 매개변수 유사도를 계산할 때 계층관계 온톨로지 개념에 위배되는 매개변수 쌍들을 배제하여 사용자가 만족할 수 있는 품질 좋은 것만 정제한다.

$$PairSim(q_i, r_j) = PairSim(q_i, r_j) \times match(q_i, r_j)$$

여기서, $match(q_i, r_j) = \begin{cases} 1 & \text{if } success \\ 0 & \text{if } fail \end{cases}$,

$$i = 1, 2, \cdots m,\ j = 1, 2, \cdots n$$

이상의 유사도 측정 방법을 적용한 Similarity 값은 우선순위 리스트를 위해 정렬될 수 있다. 이를 통해 사용자는 자신이 찾고자 하는 서비스에 대해 유사도를 기반으로 하여 순위화된 결과를 제공받을 수 있게 된다.

Ⅴ. 관련연구

시맨틱 웹 서비스에 관한 연구는 크게 두 가지 방향으로 추진되고 있다. 첫 번째 접근 방법은 전문가의 수작업으로 웹 서비스 저장소에 추가적인 시맨틱 정보를 주석처리하여 온톨로지를 구축하는 방법이다. SOAP 기반 웹 서비스에서는 OWL-S[3], WSMO[23], 그리고 SAWSDL [4] 방법이 있고, RESTful 웹 서비스에서는 SA-REST[5]와 SBWS(Semantic Bridge for Web Services)[24] 방법이

있다. 이러한 시맨틱 정보 주석처리 방법의 문제점은 전문가의 수작업으로 모든 것이 처리되어야 하므로 시간 및 인적 제약으로 인해 온톨로지를 구축하기가 어렵고, 현시점에서 웹 서비스 전체에 대한 주석을 다시 단다는 것은 거의 불가능하게 보인다.

두 번째 접근방법은 시맨틱 웹 기술을 적용하는 방법이다. 시맨틱 웹 기술은 정보를 획득하는 과정에서 현재처럼 사람이 직접 수행해야 하는 인간 중심적인 인터페이스를 컴퓨터 프로그램이 이해하고 처리할 수 있는 형태로 확장시키는 것이다. 이러한 시맨틱 웹을 실현하기 위한 온톨로지는 RDF를 기반으로 OWL과 같은 온톨로지 언어를 통해 기술된다. 관련된 프로젝트로는 유럽연합의 제7차 프레임워크 프로그램에서 지원되는 SHAPE, SOA4All, Service Web 3.0, Service-Finder 등이 있다[25]. 그러나 이 방법 또한 RDF를 구축하기 위해서는 상당히 많은 전문가의 노력이 요구되며, 실제 비즈니스 적용보다는 이론에 더 많이 치중된 연구로서 논리적 추론에 기반한 시맨틱 언어의 복잡성은 프로세스 처리나 성능 향상에 큰 부담이 될 수 있다.

이들과는 별도의 접근 방법으로써, 수작업으로 온톨로지를 구축하기가 어렵기 때문에 온톨로지 학습(learning) 방법에 의해 자동으로 온톨로지를 구축하는 방법이 연구되고 있다. Hess[26]는 웹서비스들을 자동 분류하기 위해 Naive Bayes와 SVM 머신 러닝 방법을 제안하였고, Dong[27]은 연관성이 높은 웹 서비스 매개변수들을 같은 개념으로 묶는 클러스터링 메카니즘을 제안하였다. Sabou[28]는 온톨로지 학습을 위해 프로그램 소스, 도큐멘테이션, UML 다이어그램 등 다양한 소스를 고려하였고, 자연언어처리 기법을 이용한 웹 서비스 온톨로지 학습 프레임워크를 제안하였으며, Guo[17]는 WSDL로부터 추출된 단어를 백으로 취급하지 않고 이들 간의 상관관계를 고려한 웹 서비스 매칭 방법을 제안하였다. 그러나 이러한 온톨로지 학습 방법에 관한 연구는 거의 대부분 SOAP 기반 웹 서비스를 위한 방법이었고 RESTful 웹 서비스에 관한 시도는 아직 미진한 상황이다.

VI. 결론

본 논문에서는 Open API를 개발할 때 생성되는 WSDL이나 WADL, 또는 HTML 페이지만 가지고 항목간 숨어있는 시맨틱 정보를 찾아내어 온톨로지를 자동 구축하고, 이를 이용한 유사 API 발견 방법을 제안하였다. 기존의 온톨로지 구축 방법과 다른 점은 Open API들은 일반적으로 REST, SOAP, JavaScript, XML-RPC 등 다양한 프로토콜로 제공되므로 각 프로토콜별 별도의 온톨로지를 구축하기 보다는 하나의 통합된 온톨로지 학습 방법을 통해 일괄적인 온톨로지 구축 메카니즘을 제공하는 것이다. 이로부터 REST 뿐만 아니라 SOAP을 비롯한 다양한 프로토콜에 대한 통합 검색이 가능하다.

본 논문에서 제안하는 기법들은 이전에 이미 우리가 제안하였던 SOAP 기반 온톨로지 학습 방법[11]과 RESTful 온톨로지 학습 방법[12]을 통합하여 다양한 프로토콜 상에서 일관된 처리 절차가 가능하도록 개선 발전된 기법이다. 특이한 점은 두 웹 서비스 기술의 차이점 즉, SOAP 기반 웹 서비스는 WSDL을 매개체로 서비스 능력을 개방하고자 한 기술이고, RESTful 웹 서비스는 URI를 기반으로 리소스를 개방하고자 한 기술임에도 불구하고, 메소드와 입출력 매개변수와 같은 서비스 구조의 공통 요소로부터 같은 절차로 온톨로지 정보를 추출할 수 있었고, 이를 기반으로 한 유사 웹 서비스 발견 방법을 유도할 수 있었다.

본 논문에서 제시된 스마트 매쉬업 시스템은 현재 개발 중에 있으며 여기서는 온톨로지 학습 방법과 탐색 엔진 모듈만 중점적으로 다루었다. 하지만 본 시스템은 처음부터 완전히 새롭게 개발되는 것이 아니라 본 연구실에서 이미 개발되어 있는 동적 웹 서비스 조합 시스템인 SemanticBPEL 시스템[29]을 기반으로 추가 기능들을 확장・발전시킬 예정이다. 향후 연구 과제로는, 스마트 매쉬업을 위한 Open API들의 (반)자동 조합 기법에 관한 연구가 수행될 필요가 있다.

참고문헌

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

[2] http://webservices.seekda.com

[3] OWL Services Coalition, “OWL-S: Semantic Markup for Web Services,” OWL-S White Paper, 2004.

[4] J. Kopecky, T. Vitvar, C. Bournez, and J. Farrell, “SAWSDL: Semantic Annotations for WSDL and XML Schema,” IEEE Internet Computing, Vol. 11, No. 6, 2007, pp. 60-67.

[5] A. P. Sheth, K. Gomadam, and J. Lathem, “SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups,” IEEE Internet Computing, Vol. 11, No. 6, 2007, pp. 91-94.

[6] W3C, “Web Services Description Language (WSDL) Version 2.0 Part1: Core Language,” http://www.w3c.org/TR/wsdl20/2003.

[7] http://www.w3.org/Submission/wadl/

[8] 염귀덕, “웹 서비스 품질 태스팅 프레임워크,” 디지털산업정보학회논문지, 제2권, 제2호, 2006, pp. 89-97.

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

[10] 박유미, 문애경, 유현경, 정유철, 김상기, “SOAP 기반 웹 서비스와 RESTful 웹 서비스 기술 비교,” 전자통신동향, 제25권, 제2호, 2010, pp. 112-120.

[11] 이용주, “온톨로지 학습에 의한 유사 웹 서비스 오퍼레이션 발견 방법,” 정보처리학회논문지D, 제18-D권, 제2호, 2011, pp. 133-142.

[12] Yong-Ju Lee, Chang-Su Kim, “Building Semantic Ontologies for RESTful Web Services,” In: Proceedings of the 6th International Conference on Next Generation Web Services Practices, 2010, pp. 37-40.

[13] http://www.restlet.org/

[14] R. Agrawal, T. Imielinski, and A. Swami, “Mining Association Rules between Sets of Items in Large Databases,” In: Proceedings of the 1993 ACM-SIGMOD International Conference Management of Data, 1993, pp. 207-216.

[15] D. Braga, A. Campi, S. Ceri, M. Klemetinen, and P. Lanzi, “Discovering Interesting Information in XML Data with Association Rules,” In: Proceedings of the 2003 ACM Symposium on Applied Computing, 2003, pp. 450-454.

[16] R. Agrawal and R. Srikant, “Fast Algorithm for Mining Associations Rules,” In: Proceedings of the 20th VLDB Conference, Santiage, Chile, 1994, pp. 487-499.

[17] H. Guo, A. Ivan, R. Akkiraju, and R. Goodwin, “Learning Ontologies to Improve the Quality of Automatic Web Service Matching,” In: Proceedings of IEEE International Conference on Web Services, 2007, pp. 118-125.

[18] P. Velardi, P. Fabriani, and M. Missikoff, “Using Text Processing Techniques to Automatically Enrich a Domain Ontology,” In: Proceedings of the ACM International Conference on Formal Ontology in Information Systems, 2001, pp. 270-284.

[19] http://www.xmethod.net/

[20] http://crftagger.sourceforge.net/

[21] 전인하, 문현정, 김영지, 우용태, “인적 자원 관리를 위한 사용자 온톨로지 생성 기법,” 디지털산업정보학회논문지, 제3권, 제4호, 2007, pp. 1-10.

[22] G. Salton and C. Buckley, “Term Weighting Approaches in Automatic Text Retrieval,”

Information Processing and Management, Vol. 24, No. 5, 1988, pp. 513-523.

[23] T. Vitvar, M. Zaremba, M. Moran, M. Zaremba, and D. Fensel, "SESA: Emerging Technology for Service-Centric Environment," IEEE Software, Vol. 24, No. 6, 2007, pp. 56-67.

[24] R. Battle and E. Benson, "Bridging the Semantic Web and Web 2.0 with Representational State Transfer (REST)," Journal of Web Semantics Vol. 6, 2008, pp. 61-69.

[25] Wikipedia, Semantic Web Services, http://en. wikipedia. org/wiki/Semantic_Web_Services/

[26] A. Hess and N. Kushmerick, "Learning to Attach Metadata to Web Services," In: Proceedings of the 2nd International Semantic Web Conference, 2003, pp. 258-273.

[27] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity Search for Web Services," In: Proceedings of VLDB, 2004, pp. 372-383.

[28] M. Sabou, C. Wroe, C. Goble, and H. Stuckenschmidt, "Learning Domain Ontologies for Semantic Web Service Descriptions," Journal of Web Semantics, Vol. 3, No. 4, 2005, pp. 340-465.

[29] 이용주, "반자동 웹 서비스 조합을 위한 WS-BPEL과 OWL-S의 융합 시스템," 정보처리학회논문지D, 제15-D권, 제4호, 2008, pp. 569-580.

■ 저자소개 ■

이 용 주
Lee, Yong Ju

2008년 1월~현재
경북대학교 컴퓨터정보학부 교수
1998년 8월~2007년 12월
상주대학교 컴퓨터공학과 부교수
1989년 3월~1994년 7월
삼보컴퓨터 근무
1985년 3월~1989년 2월
KIST시스템공학센터 연구원
1997년 8월 한국과학기술원 정보및통신공학과
컴퓨터공학전공(공학박사)
1985년 2월 한국과학기술원 산업공학과
정보검색전공(공학석사)

관심분야 : 웹 서비스, 정보검색
E-mail : yongju@knu.ac.kr:

논문접수일 : 2011년 7 월 26 일
수 정 일 : 2011년 8 월 17 일
게재확정일 : 2011년 8 월 25 일