키워드를 이용한 효율적인 웹서비스 및 openAPI 검색 엔진 개발1)

천동석[O1] 차승준[1], 김경옥[2], 이규철[1†]
[1]충남대학교 컴퓨터공학과
[2]한국정보통신연구원
{ikarus1004[O1], junii[1], kclee[1]}@cnu.ac.kr
kokim[2]@etri.re.kr

Development of Efficient Search Engine for Web services and openAPIs by Keyword

Dong-Suk Chun[O1], Seung-Jun Cha[1], Kyong-Ok Kim[2], Kyu-Chul Lee[1]
[1]Department of Computer Engineering, Chungnam National University
[2]Telemetics Research Group, ETRI

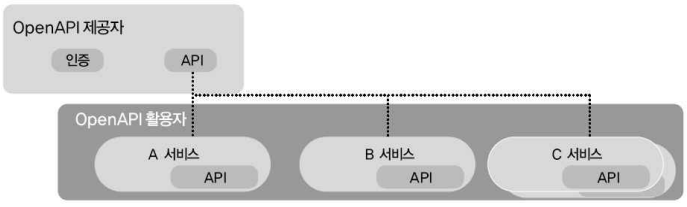
요 약

최근 많이 주목을 받고 있는 웹 2.0은 사용자의 참여, 개방, 네트워크 효과에 기반하여 누구나 데이터를 생산하고 공유할 수 있는 사용자 중심의 인터넷 환경이다. openAPI는 웹 2.0의 근본 개념인 데이터의 개방 및 공유를 구현할 수 있는 핵심 기술로 포털은 자신의 서비스를 공개한다. 하지만 기존의 웹서비스와 openAPI 검색은 효율적인 검색 방법을 제공하지 않았다. 본 논문에서는 Lee[1]의 논문에서 제공하는 효율적인 웹서비스 검색 방법을 이용하여 검색 엔진을 개발하였다. 하지만 이 연구는 웹서비스와 유사한 구조를 가지는 openAPI를 수용하지 못한다. 따라서 본 논문에서는 openAPI의 정의 및 사용법이 웹서비스와 유사하다는 점을 활용하여 openAPI의 효과적인 검색을 위한 검색기법을 개발하였다. 이러한 검색기법은 사용자가 키워드를 입력하여 키워드 기반 검색을 통해 원하는 서비스를 찾아주고, 매쉬업 서비스나 다른 openAPI와의 조합(Composition)을 위해 템플릿 기반 검색을 통해 효과적인 검색을 제공해준다.

1. 서 론

웹서비스(Web Service)는 네트워크 상에서 서로 다른 종류의 컴퓨터들 간에 상호작용을 하기 위한 소프트웨어 시스템으로 웹서비스는 서비스 지향적 분산 컴퓨팅 기술의 일종이다. 웹서비스 프로토콜 스택은 SOAP, WSDL, UDDI 등으로 이루어진다. 모든 메시징에 XML이 사용되어 상호운용성이 높다. 또한 표준화된 웹 기반 인터넷 프로토콜(HTTP, TCP/IP)을 통하여 구현되어 언제 어디서나 접근성이 좋은 장점을 가진다.

openAPI(Application Programming Interface)는 웹 2.0의 근본개념인 '데이터의 개방 및 공유'를 구현할 수 있는 핵심 기술이다. 또한 서비스, 정보, 데이터 등 언제, 어디서나, 누구나 쉽게 이용할 수 있도록 개방된 API를 의미한다. 이는 SOAP(Simple Object Access Protocol), 자바스크립트 등 웹 기술에 의해 웹 사이트들이 서로 상호작용을 할 수 있도록 하는 기술로 이루어져 있다. openAPI의 기본적인 개념은 그림 1과 같다. openAPI 제공자는 openAPI 활용자들에게 각각 인증키를 분배하고 활용자들은 그 인증키를 소유함으로써 제공자의 openAPI를 사용할 수 있게 되는 것이다.



[그림 1] openAPI의 개념

각각의 openAPI 제공자들의 서비스를 이용하여 매쉬업 서비스를 만들 수 있다. 매쉬업(Mashup)이란 여러 데이타 소스들을 가지고 한개의 웹 페이지를 구성하거나, 여러 기능을 하나의 어플리케이션에서 제공하도록 만드는 것을 말한다. 매쉬업을 이용하면 축적된 데이터 없이도 창의적인 서비스를 빠르게 만들어 볼 수 있다. openAPI는 웹서비스에서 사용되는 SOAP, REST등의 방식으로 통신을 하며, 메소드와 인자값으로 구성되어 웹서비스 구조와 유사하다.

1) 본 연구는 건설 교통부 첨단도시기술개발사업-지능형 국토정보기술혁신 사업과제의 연구비 지원(07국토정보 C05)와 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 육성, 지원사업(IITA-2008-C1090-0801-0031)의 연구 결과로 수행되었음.
† 교신저자

웹서비스는 UDDI를 검색을 위해 UDDI 저장소를 사용한다. UDDI는 웹서비스 검색을 지원하지만 한정된 범위 및 키워드 질의만 가능하다는 단점이 있다. 이를 해결하기 위해 많은 연구들이 진행되고 있다[1]. 또한 현재 openAPI는 검색을 위해서는 사용자가 직접 자신이 필요한 openAPI를 찾아야 한다. 폭발적으로 늘어나는 openAPI의 종류와 양은 사용자에게 양질의 openAPI를 제공을 하는 이점도 있지만 종류의 방대함과 openAPI 수의 증가는 사용자로 하여금 자신에게 알맞은 서비스를 찾기 어렵게 하는 문제점이 있다.

이러한 문제점을 해결하기 위해 본 논문에서는 웹 서비스와 openAPI의 구조가 유사한 것을 이용하여 웹서비스 검색에 대한 연구들을 살펴보고, 웹서비스 검색 시스템에 openAPI를 적용한 검색 시스템을 개발하였다.

개발된 시스템을 통해 사용자는 키워드를 통해 원하는 서비스를 검색(키워드 기반 검색) 할 수 있고 또한 조합하고자 하는 서비스를 검색(템플릿 기반 검색) 할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 지금까지 연구되고 있는 웹서비스 검색에 관련된 연구에 대해 살펴보고, 3장에서는 openAPI의 검색 개념에 대해서 설명한다. 4장, 5장에서는 키워드 기반 검색과 템플릿 기반 검색에 대해서 설명한다. 6장에서는 구현환경 및 개발 시스템에 대한 설명을 하고, 7장에서는 결론을 제시하며 향후 연구에 대하여 기술한다.

2. 관련연구

2.1 openAPI 현황

현재 openAPI는 400여개 이상 존재하며, 음악, 검색, 지도와 같이 지속적으로 API가 추가되는 분야가 있는 반면, 파일 공유, 지불수단 등 초기에 등록된 이후 더 이상 변화가 없는 분야와 보안, E-mail, 데이터베이스 등 최근에 API가 제공되기 시작한 분야도 있다[2]. 대표적인 openAPI로는 구글맵스(GoogleMpas), 플리커(Flickr), 아마존(Amazon), 유투브(YouTube), 야후맵스(YahooMaps), 411Sync, del.icio.us, 이베이(eBay), 야후(Yahoo), MS 버츄얼어스(VirtualEarth)등이 있다.

국외 openAPI의 활용도를 살펴보면, 구글은 매쉬업 서비스에서도 가장 많이 이용되고 있는 openAPI로서 전 세계 지도를 서비스 하고 있다. 플리커는 사용자가 직접 사진을 등록, 관리하고 공유할 수 있는 서비스를 제공하는 사이트이다. 아마존은 아마존에서 판매하는 상품 정보를 제공한다. 특히 아마존은 구매자가 아마존 웹서비스를 이용하여 구축된 사이트를 통해 상품을 구입하는 경우, 아마존은 해당 사이트에 수익의 일부를 배분하는 비즈니스 모델을 가진다.

국내에서는 네이버(Naver)와 다음(Daum)에서 제공하는 openAPI가 많이 사용된다. 네이버의 openAPI는 크게 지도 서비스와 검색 서비스로 구분된다. 그 중 다음과 비교할 때, 지도 API는 독자적인 서비스 영역을 가지고 있다. 다음은 openAPI를 통해 검색, 블로그, 디앤샵(D&Shop) 등에 대해 다양하고 재미있는 서비스 및 애플리케이션을 개발할 수 있도록 해준다[2].

2.2 웹서비스 검색

웹서비스 검색은 크게 레지스트리 기반 검색, 시맨틱 기술 정보를 이용한 검색, 유사도 기반 검색으로 분류할 수 있다.

레지스트리 기반 검색은 웹서비스 제공자와 요청자가 웹서비스에 대한 WSDL 문서와 설명 정보를 레지스트리에 등록하고, 검색하는 방법이다. 그러나 이 UDDI는 웹서비스의 등록에 대한 단일화 된 방법을 제공한다는 이점은 있지만, 검색에 있어 여러 제약점을 가진다.

시맨틱 기술 정보를 활용한 검색 방법은 서비스의 검색과 이용을 위해 웹서비스 표준 기술에 시맨틱 설명 정보를 추가하고 이의 추론을 통한 검색과 이용의 가능함을 목적으로 하고 있다. 대표적인 연구로는 OWL-S[3], WSDL-S[4], METEOR-S[5]와 같은 연구들이 존재한다. 이들 연구들은 레지스트리 기반 검색 보다 지적이며 정확한 발견을 보장하지만 다음과 같은 문제점들을 가진다. 시맨틱 정보의 추론과 처리에 많은 부하가 걸리고, 서비스 개발 시 서비스의 시맨틱 정보를 자동 생성하기가 어렵다.

레지스트리 기반 검색의 문제점과 레지스트리 기반 검색의 해결을 위한 시맨틱 기술 정보를 이용한 검색의 방법에서의 이런 문제점으로 인하여, 최근에는 유사도 기반 검색과 같이 기존의 정보 검색 기법을 도입하여 웹서비스 검색 방법을 풀고자 하는 시도가 이루어졌다. 이러한 연구들은 웹서비스 또는 서비스 내의 연산자들에 대한 유사도를 계산함으로써 가장 유사도가 높은 서비스들을 추려 검색 결과로 제공한다. Woogle[6]은 WSDL에서 정의된 operation들의 검색을 주요 목적으로 하는 웹서비스 검색 엔진이다. 이 시스템은 단어들 간의 연관성의 분석을 통해 연산자들과 이들의 파라미터를 서로 의미적으로 동일한 개념들로 클러스터링하는 기법을 이용하였다. Peng[7]은 형식 개념 분석(formal concept analysis) 기법을 이용하여 Woogle처럼 서비스들을 의미적으로 그룹화하는 기법을 제시하였다.

Lee[1]은 서비스 검색에 있어 기존 기법들과 달리 UDDI 레지스트리에 저장된 서비스 설명 정보와 WSDL 서비스 정의 모두를 활용하였으며 n-gram을 이용한 키워드 질의를 통해 부분문자열의 효율적인 검색을 보장하였고 가중치 부여와 유사도 계산 기법을 새로 고안하여 사용자는 키워드 질의를 통해 순위화 된 서비스 검색 결과를 제공 받을 수 있다.

이 연구에서는 용어 기반 역인덱스 검색의 기본 구조로 이용하고 있으므로, 가중치도 용어에 대해서 부여된다. 또한 키워드 질의를 통한 순위화는 사용자에게 알맞은 서비스를 제공한다.

Lee[1]에서는 웹서비스의 정보를 효율적으로 검색할 수 있지만 웹서비스와 유사한 구조를 가지는 openAPI를 수용하지 못한다. 따라서 본 논문에서는 앞선 연구의 웹서비스 검색 구조를 활용하여 openAPI의 저장을 가능하게 하는 저장 구조와 이를 통해 검색을 가능하게 하는

검색 엔진을 개발하였다.

3. openAPI 검색

3.1 openAPI 구조

openAPI는 XML-RPC, REST, SOAP의 웹서비스에서 사용하는 통신방법을 사용한다. XML-RPC는 원격 컴퓨터의 메소드 호출을 XML로 캡슐화하여 전달하고 결과 역시 XML로 전달을 받으며 기존 HTTP 프로토콜을 그대로 이용한다. REST은 도메인 지향 데이터를 HTTP위에서 SOAP이나 쿠키를 통한 세션 트랙킹 같은 부가적인 전송 레이어 없이, 전송하기 위한 아주 간단한 인터페이스이다. SOAP은 소프트웨어간에 메시지를 교환하는 형태의 프로토콜이다. SOAP은 확장 가능한 분산 프로토콜로 HTTP나 SMTP 등의 다양한 통신 프로토콜을 사용하여 전달이 가능하다.

openAPI는 사용되는 통신 방법에 따라 SOAP 또는 XML-RPC등 웹서비스에 균일한 파라미터 전달을 시도하는 함수형과 도메인 지향 데이터를 HTTP위에서 SOAP이나 쿠키를 통한 세션 트랙킹 같은 부가적인 전송 레이어 없이, 전송하기 위한 아주 간단한 인터페이스인 REST형으로 구분된다.

함수형 openAPI는 클래스, 클래스에 대한 설명, 메소드와 메소드에 대한 설명으로 이루어졌다. 대표적으로 구글 맵스 openAPI, 네이버 지도 openAPI등 지도 서비스들이 함수형 openAPI를 제공한다. 그림 2은 함수형 openAPI 중 네이버 지도 openAPI의 예제이다.

1. NMap

• Constructor

Constructor	Description
NMap(container[, width][, height])	새로운 지도 객체를 생성한다. container는 지도를 표시할 HTML container이며 일반적으로 DIV 요소(element)이다. width는 지도의 너비이다. height는 지도의 높이이다. 만약, width와 height의 값이 제공되지 않으면 container의 width와 height의 값을 사용한다.

• Methods

Method	Return Value	Description
getBound()	Array(left, top, right, bottom)	현재 지도의 경계(boundary) 영역 정보를 가지고 있는 배열(Array)을 반환한다. left, top, right, bottom은 각각 현재 경계 영역의 좌측,정상,우측,바닥 가장자리의 좌표값이다.
setBound(left, top, right, bottom)	none	현재 지도의 경계(boundary) 영역을 설정한다. left, top, right, bottom은 각각 경계 영역의 좌측,정상,우측,바닥 가장자리의 좌표값을 설정하는데 사용된다. 설정이 완료되면 현재 지도의 경계와 지도 크기를 이용해서 지도의 중앙점(center point)과 축적 수준(zoom level)을 설정한다.
setCenterAndZoom (point, zoom_level)	none	현재 지도의 중앙점(center point)과 축적 수준(zoom level)을 설정한다. point는 NPoint 클래스의 객체이며 zoom_level은 1~11 사이의 자연수이다. zoom_level이 1에 가까울 수록 지도가 확대된다.
setCenter(point)	none	현재 지도의 중앙점(center point)을 설정한다. point는 NPoint 클래스의 객체이다.
getCenter()	NPoint	현재 지도의 중앙점(center point)를 반환한다.

[그림 2] 네이버의 지도 openAPI의 예

REST형 openAPI는 요청 URL과 각각의 요청변수를 익스플러러 주소창에 입력하면 사용자가 지정한 XML, JSON등의 형식으로 정보를 제공해준다. 대표적으로 네이버 검색 openAPI, 야후 검색 openAPI, 플리커 openAPI등 지도외 서비스들이 REST형 openAPI를 제공한다. 그림 3은 REST형에 대한 openAPI 예제이다.

• 지식iN검색 지식iN 검색결과 API를 제공해드립니다. 다양하고 풍부한 지식iN 검색결과를 여러분의 웹사이트에 응용해보세요.

1. 요청 URL (request url)

http://openapi.naver.com/search

2. 요청 변수 (request parameter)

요청 변수	값	설명
key	string (필수)	이용 등록을 통해 받은 key 스트링을 입력합니다.
target	string (필수) : kin	서비스를 위해서는 무조건 지정해야 합니다.
query	string (필수)	검색을 원하는 질의, UTF-8 인코딩 입니다.

3. 출력 결과 필드 (response field)

필드	값	설명
rss	-	디버그를 쉽게 하고 RSS 리더기만으로 이용할 수 있게 하기 위해 만든 RSS 포맷의 컨테이너이며 그 외의 특별한 의미는 없습니다.
channel	-	검색 결과를 포함하는 컨테이너입니다. 이 안에 있는 title, link, description 등의 항목은 참고용으로 무시해도 무방합니다.
lastBuildDate	datetime	검색 결과를 생성한 시간입니다.
total	integer	검색 결과 문서의 총 개수를 의미합니다.

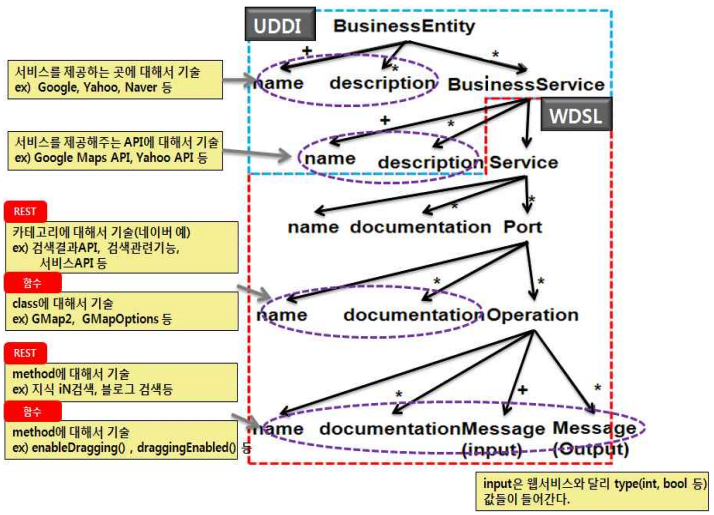
[그림 3] 네이버 지식iN검색의 예

함수형과 REST형의 반환타입은 좀 차이점이 있다. 함수형은 int, none, Boolean등 1개의 타입이 반환되는 반면 REST형은 여러 개의 출력 결과 필드가 반환된다.

3.2 웹서비스와 openAPI 저장 구조

Lee[1]에서는 웹 서비스 검색을 위해 UDDI 데이터와 WSDL 정의에서 필요한 정보만 추려 독자적인 구조의 서비스 설명 정보를 구성한다. UDDI 데이터 부분에 속하는 BusinessEntity섹션과 BusinessService섹션에는 함수형과 REST형 모두 서비스를 제공하는 곳과 서비스를 제공해주는 API에 대해서 기술하지만 WSDL 데이터 부분에 속하는 Port섹션과 Operation섹션에서는 함수형과 REST형의 구조의 차이 때문에 Lee[1]의 서비스 설명 정보에 함수형과 REST형의 openAPI 맵핑 방법의 차이를 두었다.

그림 4은 본 논문에서 사용하는 본 시스템의 저장 구조 정보를 보인다.



[그림 4] 본 시스템의 저장 구조 정보

UDDI의 BusinessEntity 섹션은 비즈니스 개체에 대한 이름, 설명, URL, 연락처와 식별 및 분류를 위한 정보를 표현하기 위한 최상위 구조이며, 하나의 businessEntity에는 여러 개의 서비스에 대한 논리정보를 가지는 BsinessService를 가질 수 있다. 본 논문에서는 BusinessEntity 섹션의 name 엘리먼트에서는 서비스를 제공해주는 제공자(예: Google, Yahoo, Naver 등)에 대해서 기술하였고, description 엘리먼트에서는 그 제공자에 대한 설명에 대해서 기술하였다.

BusinessService 섹션은 비즈니스 개체가 제공하는 서비스에 대한 논리 정보를 표현하는데 이용된다. 본 논문에서는 BusinessService 섹션의 name 엘리먼트에서는 서비스를 제공해주는 해당 API에 대해서 기술하였고, description 엘리먼트에서는 API에 대한 설명에 대해서 기술하였다.

WSDL의 Service 섹션은 UDDI의 BusinnesService의 개념과 같기 때문에 본 논문에서는 Service 섹션은 매칭시키지 않았다. 하나의 Serivce에는 여러 개의 포트에 대한 논리정보를 가지는 Port를 가질 수 있다.

Port 섹션과 Operation 섹션은 함수형과 REST형의 구조적 차이로 때문에 맵핑 방법의 차이를 두었다.

함수형에서는 Port 섹션의 name 엘리먼트에서는 클래스의 이름을 기술 하였고, documentation 엘리먼트에서는 클래스의 설명에 대해서 기술하였다. Operation 섹션의 name 엘리먼트에서는 오퍼레이션의 이름을 기술 하였고, documentation 엘리먼트에는 오퍼레이션의 설명에 대해서 기술하였다. 또한 Input, Output 엘리먼트에는 입출력 인자에 대해서 기술하였다.

REST형에서는 Port 섹션의 name 엘리먼트에서는 카테고리에 대해서 기술하였다. 예를들어 네이버에서 지식iN 검색, 블로그 검색등을 포함하는 검색결과 API, 실시간 검색어, 추천 검색어등을 포함하는 검색관련기능, 지식스폰서 API, 데스크톱 위젯 API등을 포함하는 서비스 API등이 이에 속한다. documentation 엘리먼트에서는 각각의카테고리의 설명에 대해서 기술하였다. Operation 섹션의 엘리먼트에서는 지식 iN검색, 실시간 검색어등의 오퍼레이션의 이름을 기술 하였고, documentation 엘리먼트에는 오퍼레이션의 설명에 대해서 기술하였다. 또한 Input, Output 엘리먼트에는 요청 필드 값과 응답 필드 값에 대해서 기술하였다.

그림 5은 본 논문에서 사용하는 함수형 openAPI 저장 구조의 예를 보여준다.

4. 키워드 기반 검색

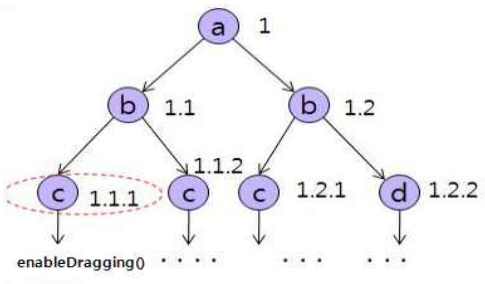
키워드 기반 검색은 사용자가 키워드를 입력하여 원하는 서비스를 찾는 것으로, 웹서비스 또는 openAPI에 대하여 서비스 검색 영역(비즈니스, 서비스, 오퍼레이션)을 설정하고 사용자가 입력받은 키워드에 대하여 유사도를 계산하여 순위화된 정보를 보여준다.

```
- <businessEntity>
    <name xml:lang="en">Google</name>
    <description xml:lang="en">Google, a popular search engine, is a tool for finding resources on
      the World Wide Web. Google scans web pages to find instances of the keywords you have
      entered in the search box.</description>
  - <businessService>
      <name xml:lang="en">Google Maps API</name>
      <description xml:lang="en">The Google Maps API lets you embed Google Maps in your own
        web pages with JavaScript. The API provides a number of utilities for manipulating
        maps (just like on the http://maps.google.com web page) and adding content to the
        map through a variety of services, allowing you to create robust maps applications on
        your website.</description>
    - <Service>
        <name />
      - <Port>
          <name>GMap2</name>
          <documentation>Instantiate class GMap2 in order to create a map. This is the
            central class in the API. Everything else is auxiliary.</documentation>
        - <Operation>
            <name>GMap2(container,opts?)</name>
            <documentation>Creates a new map inside of the given HTML container, which
              is typically a DIV element. If no set of map types is given in the optional
              argument opts.mapTypes, the default set G_DEFAULT_MAP_TYPES is used.
              If no size is given in the optional argument opts.size, then the size of the
              container is used. If opts.size is given, then the container element of the
              map is resized accordingly. See class GMapOptions.</documentation>
            <input>GMap2Request</input>
            <output>GMap2Response</output>
          </Operation>
        </Port>
      - <Port>
          <name>GInfoWindow</name>
          <documentation>GInfoWindow has no constructor. It is created by the map and
            accessed by its method GMap2.getInfoWindow().</documentation>
        - <Operation>
            <name>reset(latlng, tabs, size, offset?, selectedTab?)</name>
            <documentation>Resets the state of the info window. Each argument may be
              null and then its value will not be changed from the current
              value.</documentation>
            <input>resetRequest</input>
            <output>resetResponse</output>
          </Operation>
        </Port>
      </Service>
    </businessService>
  </businessEntity>
```

[그림 5] 함수형 openAPI 저장 구조 예제

4.1 질의의 모델링

본 시스템의 저장 구조 정보는 XML로 기술되며, 계층구조를 가지므로 질의는 다음과 같이 정의된다. UDDI에 저장된 모든 본 시스템의 저장 구조 정보에 대한 집합을 D라 했을 때, h-번째 설명정보 $d_h \in D$는 $\{t_i, l_j, o_k | t_i \in T, l_j \in L, o_k \in O\}$으로 정의된다. 여기에서 T, L, O 는 각각 D에서의 모든 용어(term)들과위치(location)들과 순서값(ordinal value)들의 집합이다. 그리고 위치 lj는 경로로 표현되며, 설명정보 d에서 노드 u의 경로란 u에서 루트까지의 경로 상에 있는 노드들의 연속적인 순열이다.



[그림 6] 순서값이 부여된 XML 문서의 예

그림 6에서 C 엘리먼트의 텍스트는 (t_i, l_j, o_k) = ('enableDragging()', /a/b/c, 1.1.1)으로 표현한다. 여기에서 순서값 o_k는 트리에서 각 노드의 순서 정보를 표현하기 위해 부여된 값으로 본 논문에서는 이 값의 표현에

듀이 넘버링 기법을 이용한다.

XML 문서에서 특정 텍스트를 포함하는 엘리먼트를 포함한 서브트리를 검색하는 질의 q는 함수 root: tree $t \rightarrow r$, 여기에서 r은 결과 트리의 루트 노드를 가지는 리스트 $\langle root(q), \{(t_i, l_j)/t_i \in T, l_j \in L\}\rangle$ 로 표현할 수 있다. 하지만, 실제 서비스 검색에서는 검색의 단위가 비즈니스, 서비스, 또는 연산자라는 개념적인 단위로 한정이 된다. 이를 위해 함수 root를 root: tree $t \rightarrow r$, r은 개념 트리 c의 루트 노드로 재 정의한다. 이에 따라 $root(q)$의 값은 business, service, operation 중 하나이어야 한다. 또한 각 개념과 대응되는 엘리먼트, 이는 해당 개념이 대표하는 서브트리의 루트 노드를 의미한다.

우리는 또한 가중치를 부여한 질의 q_w를 $\langle \text{root}(q), \{w_{ij} \cdot (t_i, l_j) | t_i \in T, l_j \in L\}\rangle$, 여기에서 w_{ij} 는 j 위치에 있는 i-번째 용어에 부여되는 가중치로 정의한다.

4.2 가중치 부여

정보 검색에서 가장 일반화된 가중치 부여 방법 기법은 TF*IDF이지만, TF*IDF는 평문을 대상으로 하므로, 계층 구조를 가지는 본 시스템의 저장 구조 정보에 대한 가중치 계산에는 효과적이지 않다. 본 논문에서는 계층 구조를 가지는 웹 본 시스템의 저장 구조 정보에서 용어 분포에 대한 가중치를 부여하기 위해 Lee[1]의 가중치 기법인 TF*IEF을 활용한다.

4.3 유사도 측정

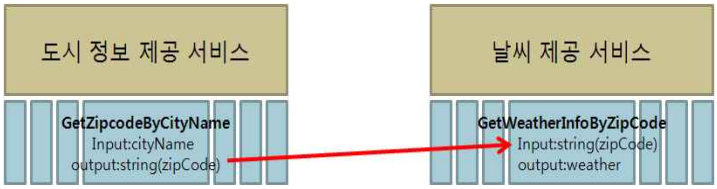
서비스 유사도의 측정을 위해 본 논문에서는 VSM을 이용한다. 즉, 질의와 본 시스템의 저장 구조 정보는 모두 각 원소가 정의 3에 따라 가중치 벡터로 처리된다. 즉, 본 시스템의 저장 구조 정보 집합 D에서 h-번째 본 시스템의 저장 구조 정보로부터 얻어지는 루트 r인 서브트리에 대한 가중치 벡터는 $\overrightarrow{v_h^r} = (w_{h11}^r, \ldots, w_{hij}^r, \ldots, w_{hmn}^r)$ 으로 정의되며, 여기에서 만약 해당 위치 j에 용어 ti가 존재하는 경우에는 $w_{hij}^r > 0$ 이며, 그렇지 않은 경우 $w_{hij}^r = 0$이 된다. 유사하게 질의어에 대한 가중치 벡터는 $\overrightarrow{v_q} = (w_{q11}, \ldots, w_{qij}, \ldots, w_{qmn})$으로 정의된다. 마지막으로 질의어에 대한 서비스 유사도는 두 벡터의 내적 계산을 통해 얻어진다.

4.4 부분문자열 일치 검색의 지원

openAPI에서 메소드 값으로 설정된 문자열은 여러 단어들이 합쳐져 합성 명사화되어 이용된다. 이 경우 역인덱스와 같은 단어-기반 인덱싱에서는 해당 문자열을 단어 단위로 분할을 시키던지 아니면, 원 문자열을 그대로 단어로써 인식하고 일반적인 부분문자열 검색을 행하는 방법을 수행해야 한다. DBMS에서는 한 스트링에서 부분문자열을 검색하기 위한 LIKE 연산을 지원을 하지만 LIKE 연산은 칼럼 값들에 대해 B-트리 인덱스 상의 노드들을 조회하면서 패턴 매칭을 수행함으로써 검색 성능의 저하를 야기할 수 있다. 이에 따라 본 논문에서는 부분문자열 매칭을 지원하기 위한 인덱스 기법인 n-그램을 이용한다.

5. 템플릿 기반 검색

템플릿 기반 검색은 조합을 위한 연결되는 서비스를 찾기 위해 서비스의 이름과 입/출력값을 이용하여 키워드 기반 검색과 같은 방법으로 가중치 부여와 유사도를 측정한다. 이때 출력값을 이용하면 연결되는 서비스의 입력값과 매칭이 되며, 입력값을 이용하면 현재 함수의 앞단에 서비스의 출력값과 매칭이 된다.



[그림 7] 템플릿 기반 검색 예

그림 7은 템플릿 기반 검색의 예로 사용자는 자신의 홈페이지에 도시별 날씨 정보를 제공해주고 싶을 때 도시 정보를 제공해주는 서비스로부터 도시의 zipCode(타입:string)를 획득하고 획득한 string(zipCode)를 입력 값으로 하는 오퍼레이션을 검색하는 예이다.

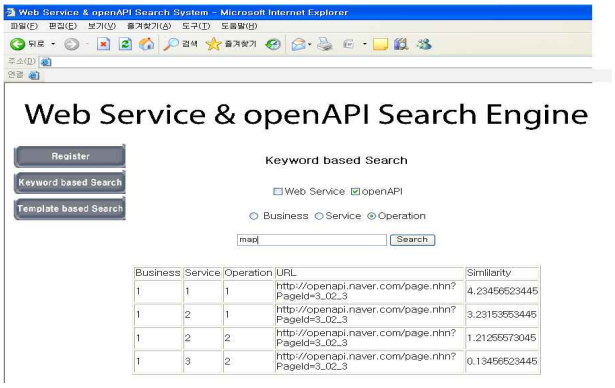
6. 시스템 구현환경

다음은 본 시스템의 구현 환경이다.

- 운영체제 : Linux Fedora 8
- 구현언어 : Java, JSP
- 데이터베이스 : Mysql

검색 시스템 구현을 위해 리눅스 플랫폼 위에 Java를 사용하였다. Java는 jdk 1.6을 이용하였고 JSP 이용을 위해 Tomcat 5.5를 사용하였다. 데이터베이스는 Mysql을 사용하였고 JSP와 JDBC를 이용하여 연동시켜 구현하였다.

7 검색 시스템 개발



[그림 8] 키워드 기반 검색 결과

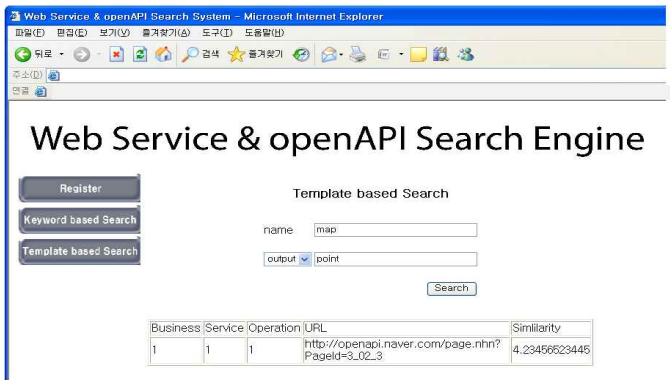
그림 8은 개발된 검색 시스템 중 키워드 기반 검색의 사용자 인터페이스이다. 사용자는 웹서비스와 openAPI를 체크박스로 지정할 수 있고, 검색 대상의 범위는 비즈니스, 서비스, 오퍼레이션 중 하나만 지정할 수 있다.

사용자가 검색대상을 선택에 따른 결과값은 다음 표 1과 같다. 비즈니스를 선택하면 사용자의 키워드가 비즈니스에 속하면 유사도 검색이 되고, 서비스를 선택하면 특정 비즈니스의 서비스에 키워드가 속하면 유사도 검색이 되고, 오퍼레이션을 선택하면 특정 비즈니스의 특정 서비스의 오퍼레이션에 키워드가 속하면 유사도 검색이 된다.

[표 1] 검색 대상에 따른 결과 필드

검색 대상	결과 필드
Business	[비즈니스] [서비스 위치] [유사도]
Service	[비즈니스] [서비스] [서비스 위치] [유사도]
Operation	[비즈니스] [서비스] [오퍼레이션] [서비스 위치] [유사도]

그림 9은 템플릿 기반 검색의 사용자 인터페이스이다. 템플릿 검색은 조합에 필요한 오퍼레이션을 찾는 검색방법이다. 대상이 되는 오퍼레이션의 이름과 입/출력 값을 통해 찾고자 하는 오퍼레이션을 포함하는 서비스를 찾을 수 있다.



[그림 9] 템플릿 기반 검색

5. 결론

본 논문에서는 웹 서비스 및 openAPI 검색을 위한 실용적이고, 효과적인 검색 기법에 대해 개발하였다.

본 논문에서는 기존 연구에서 제기되었던 UDDI 검색의 단점을 보완하기 위하여 Lee[1]의 연구를 이용하였다. 하지만 이 연구에서는 웹서비스의 정보를 효율적으로 검색할 수 있지만 웹서비스와 유사한 구조를 가지는 openAPI를 수용하지 못한다. 따라서 본 논문에서는 앞선 연구의 웹서비스 검색 구조를 활용하여 openAPI의 저장을 가능하게 하는 저장 구조와 이를 통해 검색을 가능하게 하는 검색 엔진을 개발하였다. 이는 종류의 방대함과 openAPI 수의 증가는 사용자로 하여금 자신에게 알맞은 서비스를 찾기 어렵게 하는 발견문제를 해결하기 위해 기존 정보 검색 기법을 활용하여 보다 효율적인 검색 결과를 순위를 부여하여 제공한다. 또한 웹 조합에 필요한 검색 대상을 템플릿 질의를 통해 효과적으로 찾아준다. 이를 위해 openAPI를 구성하고 있는 클래스와 메소드, 설명정보에 대해서 특정 XML 구조에 적합하도록 매핑을 시켰다.

향후 연구로는 openAPI 발견 기법들의 평가를 위 표준화된 평가 도구의 개발과 보다 다양한 조건에서의 실험, 평가가 요구된다. 또한 균일화된 openAPI 비즈니스 생성으로 자동화된 입력이 필요하다.

참고문헌

[1] Kyong-Ha Lee, Kyu-Chul Lee, To Maximize Web Service Retrieval, Convergence Information Technology ICCIT, pp 2318-2325, 2007

[2] 정찬민, 이미경, 성원경, "Open API 기술 동향", 주간 기술동향 통권 1296호, 2007

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

[4] Akkiraj, R., Farrel, J., et al., Web Service Wemantics-WSDL-S, A joint UGA-IBM Technical Note, version 1, 2005

[5] Patil, A. A., Oundhakar, S.A., et al., Meteor-S: Web Service Annotation Framework. ACM Press, NY, USA, 2004

[6] Curbera, F., et al., Unraveling the Web services web: An introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing 6(2) pp. 86-94, 2002

[7] Peng, D., Huang S., Wang, X., Zhou, A., Concept-based retrieval of alternate Web services, Proceedings of the 10th Conference on DASFAA(Database Systems for Advanced Applications) pp. 359-371, 2005