키워드 질의를 이용한 순위화된 웹 서비스 검색 기법

Ranked Web Service Retrieval by Keyword Search

이경하(Kyong-Ha Lee)*, 이규철(Kyu-Chul Lee)**, 김경옥(Kyong-Ok Kim)***

초 록

오늘날 XML 웹 서비스의 보급에 따라 많은 양의 서비스로부터 적합한 서비스들을 발견해 내는 문제는 중요한 문제로 부각되고 있다[7, 24]. 본 논문에서는 의미론적인 서비스 발견을 행하는 기존 연구와 다르게 구문적인 방법으로 서비스 발견 문제에 접근한다. 우리는 웹 서비스 발견의 문제를 레지스트리에 저장된 웹 서비스 설명 정보들에 대한 정보 검색의 문제로 인식하고, 웹 서비스와 질의의 모델링과 유사도 측정을 통한 검색 기법을 고안하였다. 이 서비스 설명정보는 특성 구조를 가지는 XML 문서이며, 본 논문에서는 이를 대상으로 한 키워드 질의를 통해 서비스 발견을 수행한다. 본 논문의 특징은 다음과 같다. 사용자는 키워드 질의를 통해 순위화된 서비스 검색 결과를 제공받을 수 있다. 서비스 검색에 있어 기존 기법들과 달리 UDDI 레지스트리에 저장된 서비스 설명 정보와 WSDL 서비스 정의 모두를 활용한다. 또한 시스템은 기존 관계형 DBMS의 기능을 이용하여 구현됨으로써 의미적인 서비스 발견에 비해 구현이 용이하고, DBMS의 여러 장점을 취할 수 있다.

ABSTRACT

The efficient discovery of services from a large scale collection of services has become an important issue[7, 24]. We studied a syntactic method for Web service discovery, rather than a semantic method. We regarded a service discovery as a retrieval problem on the proprietary XML formats, which were service descriptions in a registry DB. We modeled services and queries as probabilistic values and devised similarity-based retrieval techniques. The benefits of our way are follows. First, our system supports ranked service retrieval by keyword search. Second, we considers both of UDDI data and WSDL definitions of services amid query evaluation time. Last, our technique can be easily implemented on the off-the-shelf DBMS and also utilize good features of DBMS maintenance.

키워드 : XML 웹 서비스, 서비스 발견, 정보 검색
XML Web services, Service discovery, and Information Retrieval

---

본 연구는 건설교통부 첨단도시기술개발사업 지능형국토정보기술혁신 사업과제의 연구비지원(07국토정보 C05)와 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 육성, 지원 사업(IITA-2008-C1090-0801-0031)의 연구 결과로 수행되었음.

* 충남대학교 소프트웨어연구소 전임연구원
** 교신 저자, 충남대학교 공과대학 컴퓨터공학과 교수
*** 한국전자통신연구원 텔레매틱스연구그룹 공간정보연구팀장

1. 서 론

서비스 발견(service discovery)은 인터넷 상에 분산되어 있는 웹 서비스들을 조합하여 특정 비즈니스 프로세스를 구성하기 위해 요구되는 중요한 과정이다. 서비스 발견에 관한 기존 연구들은 크게 구문(syntactic)적 방법과 의미적(semantic) 방법으로 구분할 수 있다. 구문적 방법은 서비스 설명정보(description)가 가지는 구문적 정보(단어, 문장 등)를 활용하여 서비스를 발견하는 방법이며, 구문적 정보를 활용한 문서 검색을 수행하는 기존 정보 검색이 이에 해당한다. 이 때문에 서비스 발견의 구문적 방법을 서비스 검색(service retrieval)이라고도 한다. 의미적 방법은 웹 서비스 발견을 위해서 미리 정의된 개념들과 이들 개념 간의 관계성을 활용함으로써 구문적 방법들에 비해 보다 지적인(intelligent) 서비스 발견을 목표로 한다. 하지만, 의미적 방법은 여러 문제점들이 아직 존재하여 실제 응용 환경에 도입할 수 있을 정도로 충분히 성숙하지 않은 것으로 평가된다[20]. 또한 의미적 방법 또한 검색에 관해 기본적으로 구문적 방법에 의존적일 수밖에 없다. 이 때문에 서비스 발견을 위해서는 효과적인 구문적 방법이 우선적으로 요구된다. 하지만, 현재의 구문적 서비스 발견은 그 효용성에 있어 여러 가지 문제점을 가진다. 일례로 레지스트리 방식인 UDDI[7]는 질의 범위와 그 능력에 있어 많은 제약을 가지며, 평문을 대상으로 한 정보 검색 기법을 그대로 도입하는 경우 또한 구조화된 서비스 설명정보의 검색에 부적합하다.

위 문제점들의 해결을 위해 본 논문에서는 효과적인 서비스 검색 기법을 제안한다. 제안하는 검색 기법은 구조화된 서비스 설명정보의 특성을 고려하여, 서비스 간의 유사성을 평가하고 이를 통해 서비스를 순위화하는 특징을 가진다. 또한, 종래의 DBMS를 이용하여 구현이 용이하며, 기존의 정보 검색 기법을 그대로 적용하였을 때보다 우수한 검색 성능을 보인다.

2. 관련 연구

서비스 발견에 관한 기존의 연구는 구문적 방법과 의미적 방법으로 구분될 수 있다. 서비스 발견과 관련한 다른 분류들은[14, 20]과 같은 문헌에서 찾아볼 수 있다. 의미적 방법이란 시멘틱 웹[2] 기술의 도입을 통해 서비스의 발견을 위해 의미적 주해(semantic annotation)를 활용하는 방법이다. 이 방법에서 의미(semantics)란 상대적으로 구문적인 서비스의 설명정보에 추가되며, 서비스들은 이렇게 주해된 의미들로부터 추론을 통해 발견된다. 서비스 발견을 위한 의미적 주해와 관련한 대표적인 연구로는 OWL-S[14], WSDL-S[1], METEOR-S[16]와 같은 연구들이 존재한다. 이들 연구들은 구문적 방법보다 지적이며 정확한 발견을 보장하지만 다음의 문제점들을 가진다. 첫 번째 문제는 어떻게 이들 의미적 주해들을 자동으로 생성할 수 있는가하는 문제이다. 두 번째는 많은 양의 의미적 주해를 가지고, 확장성(scalability) 있는 추론을 어떻게 지원할 것인가 이며, 세 번째로 여러 사용자들이 동일한 서비스에 대해

서로 다른 의미들을 각기 정했을 경우 이들을 어떻게 합병 또는 통합 관리하는가 하는 문제가 존재한다. 이러한 문제들로 인하여 의미적 서비스 발견 기법은 아직 실제 응용 환경에 도입되기에는 무리가 있다고 평가된다[20].

이와 반대로, 구문적 방법은 구문에 대한 통계적 유사도를 이용하며, 전통적인 정보 검색에서 출발한다. 구문적 방법은 의미적 방법과 비교하여 질의 처리 과정에서 사람의 개입이 크게 요구되지 않고, 오랜 시간 검증된 기법이라는 이점을 가진다. Zhuge[24]은 서비스 간의 특별화(specialization) 관계성을 평가함으로써 UDDI 레지스트리의 질의 기능을 확장하였으며, [17, 19, 21]과 같은 연구들에서는 VSM(Vector Space Model)[12]을 이용한 문서 순위화 기법이 서비스 검색에서도 유용할 수 있음을 보였다. Zhuang[23]은 코퍼스에 기반을 둔 유사도 계산을 통하여 의미적으로 유사한 WSDL 파일들을 식별할 수 있는 기법을 제안하였다. Woogle[7]은 WSDL에서 정의된 operation들의 검색을 주요 목적으로 하는 웹 서비스 검색 엔진이다. 이 시스템은 단어들 간의 연관성의 분석을 통해 연산자들과 이들의 파라미터들을 서로 의미적으로 동일한 개념들로 클러스터링하는 기법을 이용하였다. Peng[18]은 형식 개념 분석(formal concept analysis) 기법을 이용하여 Woogle처럼 서비스들을 의미적으로 그룹화하는 기법을 제시하였다. 기존 연구들에 대한 간략한 조사와 평가는 [11]에서 다루고 있다. 하지만 기존 연구들은 대체로 평문을 대상으로 하는 전통적인 정보 검색 기법을 그대로 활용함으로써 XML 문법으로 표현되고, 구조화된 서비스 설명 정보에 대한 순위화에는 적절하지 않다. 또한 WSDL 문서에서 정의된 연산자만을 검색의 대상으로 함으로써[8, 18], 실제 서비스 검색에 있어서 다양한 개념 대상에 대한 질의를 지원하지 못하는 문제가 있다. 기존 구문적 방법의 또 다른 문제는 질의 처리 시 부분문자열 일치의 처리 방법에 관한 문제이다. 정보 검색에서 널리 사용되는 역 인덱스와 같은 자료 구조는 단어-기반 인덱싱(word-based indexing)으로 각 용어(term)는 독자적인 의미를 가지며, 각기 따로 인덱싱 된다. 하지만, WSDL 문서에서 name 애트리뷰트는 NMTOKEN[4] 타입으로 공백문자를 허용하지 않으며 이에 따라 이 애트리뷰트의 값은 복합명사(compound name)화되는 특징을 가진다(예 : getWeatherInfoByZipcode). 이 애트리뷰트 값, 스트링은 용어 인덱스의 구축을 위해 적절히 분할되어야 하지만, 이로 야기될 수 있는 성능 상의 문제를 고려하지 않고 있다.

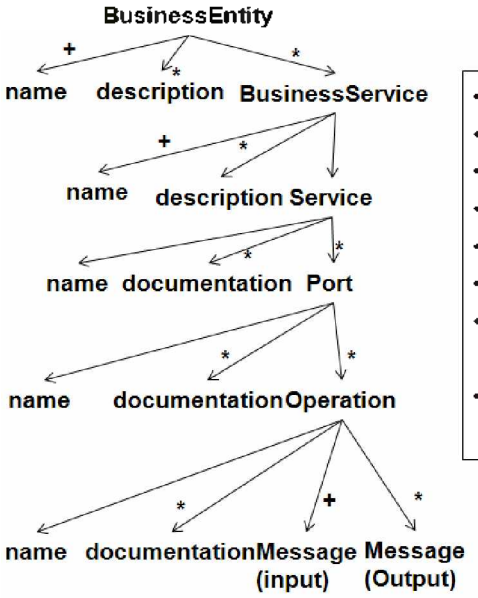
최근에는 XPath, XQuery와 같은 XML 질의 언어의 이용 시 XML 문서의 구조를 이해해야 한다는 부담을 덜고자, XML 문서에 대한 키워드 검색에 관한 연구가 진행되어왔다. XRANK[9]는 Google에서의 PageRank 기법을 엘리먼트 단위로 적용시켜 엘리먼트 단위 순위화 방법을 제시하였다. 그리고 DIL(Dewey Inverted List)이라는 듀이 넘버링 기법을 도입한 역리스트 구조를 이용하여 특정 키워드를 포함하는 노드의 탐색을 지원한다. 특히 여러 키워드가 각기 다른 노드에 걸쳐 분포할 경우 질의 키워드들 모두를 포함하는 최소 공통 조상(LCA : Lowest Common Ancestor) 노드를 구하는 기

법도 제시하였다. XSEarch[6]는 (레이블 : 키워드) 형태의 질의를 지원하며, 벡터 공간 모델을 이용한 랭킹 기법을 제시하였다. XKSearch[22]는 XML 원소들을 B^+-트리 구조로 색인하고, IL(Indexed Lookup eager) 알고리즘을 이용하여 최소 공통 조상 노드를 찾을 때, 낮은 빈도의 키워드를 포함한 노드들을 기준으로 최소 공통 조상 노드들을 찾음으로써 높은 빈도의 키워드를 포함한 노드들 전체가 읽히는 것을 방지함으로써 성능 향상을 꾀하였다. 하지만 이들 XML 문서에 대한 키워드 질의는 일반적인 XML 문서를 대상으로 특정 키워드들을 포함하는 노드 또는 최소 공통 조상 노드의 추출에 목표를 맞추고 있어, 본 논문의 특정 형식의 XML 문서인 웹 서비스 설명정보에 대하여 개념적으로 구분되는 키워드 질의를 수행하는 방법과는 차이가 있다.

3. 서비스 검색 개념

3.1 서비스 설명 정보와 질의의 모델링

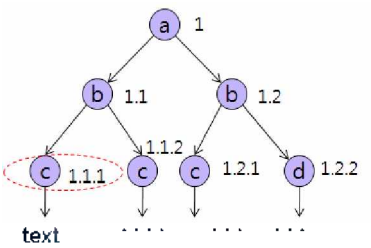
이 장에서는 서비스 검색을 위한 본 논문에서의 검색 모델을 설명한다. 서비스 검색에서 검색의 대상이 되는 문서는 평문(plain text)이 아닌 서비스 설명정보(service description)이다. 현재 서비스 설명정보는 UDDI 데이터[7]와 WSDL 정의로 나뉘어 표현된다. 본 논문에서는 이들로부터 서비스 검색에 필요한 정보만을 추려 독자적인 구조의 서비스 설명정보를 먼저 작성한다. <그림 1>은 본 논문에서 사용하는 서비스 설명 정보의 문서 구조와 경로들을 보인다. 또한 이 서비스 설명정보의 생성을 위한 알고리즘은 <표 1>에서 보인다.



• /BusinessEntity	PathId: 1
•/BusinessEntity/name	PathId: 2
•/BusinessEntity/description	PathId: 3
•/BusinessEntity/BusinessService	PathId: 4
•/BusinessEntity/BusinessService/name	PathId 5
•/BusinessEntity/BusinessService/description	PathId:6
•/BusinessEntity/BusinessService/service	PathId:7
. . .	
•/BusinessEntity/BusinessService/Service/ Operation/Output	PathId: 17

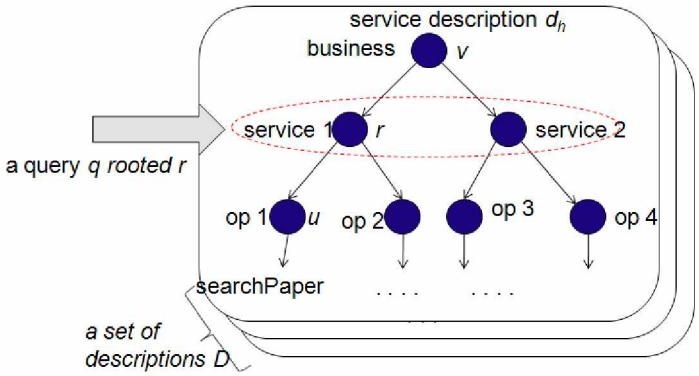
〈그림 1〉 서비스 설명정보 d_h의 문서 구조와 경로들

기존 정보 검색에서는 문서와 질의 모두 비구조적인 용어(unstructured term)들의 리스트 $\{t_i \mid 1 \le i \le n\}$, 여기에서 t_i는 리스트에서의 각 용어로 모델링되며, 질의 리스트의 용어는 ∧, ∨, ¬의 부울 연산자와 결합될 수 있다. 반면에 서비스 설명정보는 XML로 기술되며, 계층 구조를 가지므로 이 정의는 다음과 같이 확장된다. UDDI에 저장된 모든 서비스 설명정보에 대한 집합을 D라 했을 때, h-번째 설명정보 $d_h \in D$는 $\{t_i, l_j, o_k \mid t_i \in T, l_j \in L, o_k \in O\}$으로 정의된다. 여기에서 T, L, O는 각각 D에서의 모든 용어(term)들과 위치(location)들과 순서값(ordinal value)들의 집합이다. 그리고 위치 l_j는 경로로 표현되며, 설명정보 d에서 노드 u의 경로란 u에서 루트까지의 경로 상에 있는 노드들의 연속적인 순열이다. 예 : <그림 2>에서 C 엘리먼트의 텍스트는 (t_i, l_j, o_k) = ('*text*', */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 뉴이 넘버링 기법[3]을 이용한다.



〈그림 2〉 순서값이 부여된 XML 문서의 예

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 $<root(q), \{(t_i, l_j) \mid t_i \in T, l_j \in L\}>$로 표현할 수 있다. 예를 들어 텍스트 값이 '*text*'인 c 엘리먼트를 가지는 b 엘리먼트를 검색하기 위한 질의는 <b, {'*text*', '*/a/b/c*'}>로 표기되며, 이 질의는 XPath 질의 */a/b[c 'text']*와 의미적으로 동일하다. 하지만, 전체 서비스 검색에서는 검색의 단위가 비즈니스, 서비스, 또는 연산자라는 개념적인 단위로 한정이 된다. 이는 UDDI와 WSDL에서 분류해놓은 서비스 구성요소들의 개념에서 유래한다. 이를 위해 함수 root를 root : tree $t \rightarrow$



〈그림 3〉 서비스 설명정보 d_h의 논리적 구조와 서비스를 검색하는 질의 q

r, r은 개념 트리 c의 루트 노드로 재 정의한다. 여기에서 개념 트리는 <그림 3>에서 보이는 바와 같이 비즈니스, 서비스, 연산자 노드로 구성되는 본 논문에서의 하나의 서비스 설명정보이다. 이에 따라 $root(q)$의 값은 *business*, *service*, *operation* 중 하나이어야 한다. 예를 들어 '*DBLP*'라는 이름의 웹 서비스를 찾기 위한 질의는 <service, {'DBLP', /BusinessEntity/BusinessService/name'}>로 표기된다.

또한 각 개념과 대응되는 엘리먼트, 이는 해당 개념이 대표하는 서브트리의 루트 노드를 의미한다. 가령, *business* 개념의 경우에는 <businessEntity> 엘리먼트와 대응이 되며 PathId 1~3의 엘리먼트들이 이 개념에 해당한다. *service* 개념은 <businessService>를 루트로 하는 서브 트리로 PathId 4~12, *operation* 개념의 경우에는 <operation> 엘리먼트를 루트로 하는 서브 트리로 PathId 13~17까지의 엘리먼트들을 포함한다.

실제 구현에서는 경로 문자열의 반복 저장을 통한 저장 공간상의 낭비를 막기 위해 경로 문자열 대신에 각 경로마다 유일하게 할당된 PathId를 이용한다. 본 논문에서의 서비스 설명정보는 <그림 1>에서와 같은 문서 구조를 가지는 XML 문서로 지정된 17개의 PathId 값 또한 고정되어, 경로 문자열이 변경되지 않는다.

우리는 또한 가중치를 부여한 질의 q_w를 <root(q), $\{w_{ij} \cdot (t_i, l_j) \mid t_i \in T, l_j \in L\}$>, 여기에서 w_{ij}는 j 위치에 있는 i 번째 용어에 부여되는 가중치로 정의한다. 만약 사용자가 'DBLP' 서비스를 찾고자 하는 경우, 해당 서비스는 <name>이나 <description> 엘리먼트의 텍스트 값으로 이 'DBLP' 텍스트를 포함하고 있을 것이다. 경험적으로 <name>의 텍스트 값으로 해당 스트링이 위치하는 경우가 <description>의 경우보다 중요하게 인식되어야 한다. 이 경우 용어의 위치별로 질의 용어에 서로 다른 가중치를 부여하는 것이 보다 효과적이다. 이에 따라 이 예제 질의는 q_w <*service*, {0.7 · ('DBLP', 5), 0.3 · ('DBLP', 6)}와 같이 각기 다른 가중치를 부여받을 수 있으며, 실제 구현에서도 이와 같이 j 위치별로 다른 질의 가중치를 부여한다.

3.2 가중치 부여와 유사도 계산 기법

TF * IDF는 정보 검색에서 가장 일반화된 가중치 부여 기법이다. 하지만, TF * IDF는 평문을 대상으로 하므로, 계층 구조를 가지는 서비스 설명정보에 대한 가중치 계산에는 효과적이지 않다. XSearch[6]에서는 XML 문서에 대한 TF * IDF의 적용을 다루었으나, (레이블 : 키워드) 질의에 대한 결과의 순위화를 위한 기법으로, 본 논문에서 필요로 하는 특정 경로 상에 위치한 키워드에 대하여 서브 트리별로 달라지는 순위화 기법과는 차이가 있다. 본 논문에서는 계층 구조를 가지는 웹 서비스 설명정보에서 용어 분포에 대한 가중치를 부여하기 위해 다음과 같은 가중치 기법을 고안하여 활용한다.

- ***정의 1. 용어 빈도*** : t_i를 i-번째 용어라 하고, $freq_{ij}(d_h)$를 서비스 설명정보 d_h에서 용어 t_i가 위치 j에서의 출현 횟수라 하자. 또한 $maxfreq_j(d_h)$를 주어진 서비스 설명정보 d_h에서 위치 j에서 가장 많이 출현한 용어의 출현 횟수라고 하면,

d_h에서 위치 j에서의 용어 t_i의 용어 빈도 TF_{hij}는 다음과 같이 정의한다.

$$TF_{hij} = \frac{freq_{ij}(d_h)}{\max freq_j(d_h)} \quad (1)$$

이기에서 유의할 점은 용어 빈도가 계산되는 단위는 해당 용어를 포함하는 노드 단위가 아니라 경로 단위라는 점이다. UDDI에서 서비스나 비즈니스의 이름은 하나 이상의 <name> 엘리먼트를 통해 여러 번 표현될 수 있기 때문이다. 예를 들어, <name> t_1 t_2 </name>에서의 용어들과 <name> t_1 </name> <name> t_2 </name>있다고 하자. 첫 번째는 한 노드 안에 두 개의 용어 t_1, t_2가 존재하는 경우이고, 두 번째는 각 용어 t_1, t_2가 서로 다른 노드에 속하는 경우이다. 이 때 각 용어들의 용어 빈도는 서로 동일하다. 둘의 위치, 즉 경로 정보가 같기 때문이다.

역문헌빈도(IDF)는 해당 용어가 문서 집합 내에서 빈번하지 않는 경우에 보다 높은 가중치를 주기 위해 이용한다. 하지만, XML 문서에서는 모든 용어는 특정 문서 구조를 구성하는 엘리먼트들의 텍스트로 존재하며, 질의에 의해 검색되는 문헌의 범위가 가변적일 수 있다는 특징이 있다. 즉, 질의의 *root*(q)의 값에 따라 검색되는 문헌의 범위, 즉 서브 트리의 루트는 <그림 1>의 *business*이거나, *service*, *operation*일 수도 있다. 이는 검색되는 문헌의 수가 질의의 *root*(q)에 따라 가변적일 수 있음을 나타내며, 이에 따라 본 논문에서는 다음과 같이 역 엘리먼트 빈도(IEF: Inverse Element Frequency)를 고안하여 이용한다.

- ***정의 2. 역 엘리먼트 빈도*** : 질의 결과가 r이 루트인 서브트리일 때, 위치 j에서 용어 t_i에 대한 역 엘리먼트 빈도 IEF_{ij}^{r}는 다음과 같이 정의된다.

$$IEF_{ij}^{r} = \log\frac{N^{r}}{n_{ij}^{r}} + 1 \quad (2)$$

여기에서 n_{ij}^{r}는 용어 t_i를 위치 j에 포함하는, r이 루트인 서브트리의 수이며, N^{r}은 서비스 설명정보 집합 D에서 r이 루트인 서브트리의 수이며, 아래와 같이 계산된다.

$$N^{r} \begin{cases} \sum_{h,k}\{(d_h, t_j, o_k) \mid j = 13\} & \text{iff } r \text{ is an operation,} \\ \sum_{h,k}\{(d_h, t_j, o_k) \mid j = 4\} & \text{iff } r \text{ is a service} \\ |d_h| & \text{iff } r \text{ is a business} \end{cases} \quad (3)$$

여기에서 pathId $j = 4$는 루트에서 엘리먼트 <businessService>로의 경로에 대응되며, $j = 13$은 루트에서 <operation>으로의 경로에 대응된다. *root*(q)가 *business*인 경우 서비스 설명정보의 수는 <businessEntity> 엘리먼트의 수와 동일하다. 이는 모든 서비스 설명정보의 루트는 <businessEntity>이기 때문이다. 이제 이 TF와 IEF를 통해 계산되는 용어 가중치(term weight)는 다음과 같이 정의된다.

- ***정의 3. 용어 가중치*** : 질의 결과가 r이 루트인 서브 트리일 때, 서비스 설명정보 d_h에서 j 위치의 용어 t_i의 용어 가중치는 다음과 같이 정의된다.

$$w^r_{hij} = TF_{hij} \cdot IEF^r_{ij} \quad (4)$$

서비스 유사도의 측정을 위해 본 논문에서는 VSM[12]을 이용한다. 즉, 질의와 서비스 설명정보는 모두 각 원소가 정의 3의 용어 가중치 계산에 따라 가중치 벡터(weighted vector)로 처리된다. 즉, 서비스 설명정보 집합 D에서 h-번째 서비스 설명정보로부터 얻어지는 루트 r인 서브트리에 대한 가중치 벡터는 $\overrightarrow{v^r_h} = (w^r_{h11}, \cdots, w^r_{hij}, \cdots, w^r_{hmn})$으로 정의되며, 이기에서 만약 해당 위치 j에 용어 t_i가 존재하는 경우에는 $w^r_{hij} > 0$이며, 그렇지 않은 경우 $w^r_{hij} = 0$이 된다. 유사하게 질의어에 대한 가중치 벡터는 $\overrightarrow{v_q} = (w_{q11}, \cdots, w_{qij}, \cdots, w_{qmn})$으로 정의된다. 마지막으로 질의어에 대한 서비스 유사도는 두 벡터의 내적 계산을 통해 얻어지는데, 그 계산법은 다음과 같다.

- ***정의 4. 서비스 유사도*** : 질의어 q에 의해 선택되는, 루트가 r인 h-번째 서비스 설명정보의 서브 트리 d^r_h의 질의어 q에 대한 유사도는 다음과 같이 정의된다.

$$sim(d^r_h, q) = \frac{\overrightarrow{v^r_h} \cdot \overrightarrow{v_q}}{\left|\overrightarrow{v^r_h}\right| \cdot \left|\overrightarrow{v_q}\right|} = \frac{\sum_{i,j=1}^{t} (w^r_{hij} \cdot w_{qij})}{\sqrt{\sum_{i,j=1}^{t} (w^r_{hij})^2} \cdot \sqrt{\sum_{i,j=1}^{t} (w_{qij})^2}} \quad (5)$$

위의 서비스 유사도의 계산을 위해서는 $\overrightarrow{v^r_h}$이 질의 처리 시점(query time)에 한 서비스 설명정보에 대한 벡터 $\overrightarrow{v_h}$로부터 계산되어야 한다. 왜냐하면, 질의 q의 $root(q)$에 따라 한 서비스 설명정보로부터 다양한 서브트리가 추출될 수 있기 때문이다. 이에 따라 DB에 저장된 원래의 서비스 설명정보로부터 어떻게 $\overrightarrow{v^r_h}$을 빠르게 추출하느냐가 성능 상의 중요 지표가 될 수 있다.

서비스 유사도의 계산에서는 개념 관계에 따라 즉 사용자의 질의 범위 r에 속하지 않는 트리 상의 노드라 하더라도 고려해야할 필요성이 존재한다. 예를 들어 <그림 3>에서, 한 사용자가 'paper search' 서비스를 검색하고자 하여 다음과 같은 질의 q = <*service*, {('*paper*', *5*), ('*search*', *5*), ('*paper*', *6*), ('*search*', *6*)}>를 작성하였다고 하자. 이 질의를 그대로 처리한다면, 서비스 설명집합 D에서 businessService의 자식인 name, description 엘리먼트가 paper, search 용어 둘 다를 포함하지 않는 서비스들은 검색이 되지 않을 것이다. 왜냐하면, 질의의 범위가 r = service로 서비스로 한정되었기 때문이다. 하지만, 일반적인 경우 질의어와 매우 유사한 연산자들을 포함한 서비스는 그 이름이 비슷하지 않아도 검색하고자 하는 서비스일 수 있다. 반대로 비즈니스의 유사도가 질의어와 유사하다면, 해당 비즈니스는 사용자가 원하는 서비스를 가지고 있을 확률이 있다. 즉, 질의 범위를 특정 개념으로 한정한다 하더라도, 상/하위 개념의 유사도에 따라 질의 범위의 대상의 유사도는 영향을 받을 수 있어야 한다. 상/하위 개념의 유사도의 전파를 위해 본 논문에서는 식 (5)를 다음과 같이 확장한다.

- ***정의 5. 확장된 서비스 유사도*** : 상/하위 개념 간 서비스 유사도의 전파를 위

해 확장된 질의어 q와 서비스 설명정보 d_h간의 유사도는 다음과 같이 계산한다.

$$sim^{\Uparrow}(d_h^r, q) = (1-\alpha-\beta) \cdot sim(d_h^r, q) \quad (6)$$
$$+\alpha \cdot \sum_{(r,u)\in CE} \frac{sim(d_h^u, q)}{N_u}$$
$$+\beta \cdot sim(d_h^v, q)$$

여기에서 CE는 <그림 3>에서 보이는 바와 같이 개념 간 포함 관계의 집합 $\{(r, u), (v, r)\}$이며, 이 집합의 원소 (r, u)는 u가 r의 하위 개념임을 의미한다. 마찬가지로 v는 질의의 범위 r에 의해 가리켜지는 개념의 상위 개념이다. 그리고 N_u는 개념 u의 집합의 크기를 의미한다. α와 β는 각각 상위와 하위 전파 계수로 하위 개념의 유사도를 상위로 얼마만큼 전파하거나 또는 그 반대로 전파할지를 결정한다. 여기에서 $\alpha + \beta \leq 1$이다.

이전의 예제 질의 q <*service*, {('*paper*', 5), ('*search*', 5), ('*paper*', 6), ('*search*', 6)}>의 경우 r *service*이며, 이에 따라 u *operation*, v *business*이 된다. 이 식을 적용하면 *operation*에 대해 *paper*, *search* 용어에 대한 유사도를 계산하고, 이를 평균화된 값으로 원래 질의의 유사도에 α만큼 전파시킨다. 반대로, *business*에 대해 paper, search 용어에 대한 유사도를 계산한 값은 β만큼 전파된다. r = *business*인 경우에는 u *service* 이며 v에 대한 유사도는 계산되지 않는다. r = *operation*인 경우 v = *service*이며 u에 대한 유사도는 계산되지 않는다.

3.3 부분문자열 일치 검색의 지원

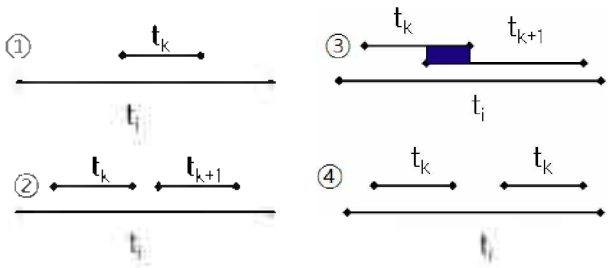
제 2장에서 언급한 바와 같이 WSDL 문서에서 애트리뷰트의 값으로 설정된 문자열은 여러 단어들이 서로 합쳐져 복합 명사화되어 이용된다. 이 경우 역 인덱스와 같은 단어 기반 인덱싱에서는 해당 문자열을 단어 단위로 분할을 시키던지 아니면, 원 문자열을 그대로 단어로써 인식하고 일반적인 부분문자열 검색을 행하는 방법을 수행해야 한다. DBMS에서는 한 스트링에서 부분문자열을 검색하기 위한 LIKE 연산을 지원하므로 이를 이용하는 것도 한 가지 방법이 된다. 하지만 LIKE 연산은 칼럼 값들에 대해 B-트리 인덱스 상의 노드들을 조회하면서 패턴 매칭을 수행함으로써 검색 성능의 저하를 야기할 수 있다. 이의 해결을 위해서 한 스트링으로부터 단어들을 구분해 내는 형태소-기반 인덱싱(morpheme based indexing)이 존재하지만, 계산 비용이 비싸고[1], 복잡한 언어적 지식을 요구하며, 잘못된 단어 추출도 행할 수 있는 문제점을 가진다. 이에 따라 본 논문에서는 부분문자열 매칭을 지원하기 위한 인덱스 기법[15]인 n-그램을 이용한다. 본 논문에서 여러 부분문자열 검색을 위한 자료 구조 중 n-그램을 선택한 이유는 크게 3가지로 첫째로 n-그램은 DBMS 내부의 변경 없이 부분문자열의 효율적인 검색을 보장할 수 있으며[10], 둘째로는 Lee 등의 연구[13]에서와 같이 복합명사가 많은 한글 같은

1) 형태소 기반 인덱싱은 한 스트링에서 추출될 수 있는 부분 문자열들에 대한 사전과의 비교 작업이 요구되며, 이 때 계산 비용은 n 크기의 스트링에 대해 $sum(n)$개 만큼의 부분 문자열이 비교되어야 하는 만큼 그 비용이 크다고 알려져 있다.

아시안 언어에서 단어 기반 인덱스와 함께 쓰여 효과적으로 부분문자열 검색을 지원할 수 있기 때문이다. WSDL과 같은 XML 문서는 모든 엘리먼트에서 xml:lang이라 불리는 다국어 코드를 입력하기 위한 특별 애트리뷰트를 가지기 때문에 이에 대한 처리가 필요하며, n-그램은 각 언어에 대한 지식이 없이도 부분 문자열 지원을 할 수 있는 특징을 가진다. 셋째로 n-그램은 문자열에 포함될 가능성이 있는 오타에 대해 견고하고(error-tolerant), 근사 질의(approximate string) 또한 지원할 수 있는 특징이 있다.

n-그램은 원 문자열에 대한 길이가 n인 부분 문자열들의 집합으로 길이가 n보다 같거나 큰 부분 문자열의 검색과 근사 질의를 지원한다. Σ를 크기가 |Σ|인 알파벳들의 유한 집합이라 하고, $o \in \Sigma^*$를 길이가 n인 문자열이라 할 때, 이 문자열의 한 n-그램은 $o[i, \cdots, j]$, $1 \le i \le j \le n$으로 표기되며, 원 문자열 o의 i번째 위치에서 시작하는 길이 $j-i+1$인 부분 문자열이다. 문자열 o에 대한 n-그램들은 길이가 n인 슬라이딩 윈도우를 o에 대해 전진시켜가면서 총 $|o|-n+1$개만큼 생성된다. 예를 들어, 'searchPaper'에 대해 $n=3$인 n-그램들은 (1, sea), (2, ear), (3, arc), (4, rch), (5, chp), (6, hpa), (7, pap), (8, ape), (9, per)이 된다. 여기에서 대소문자는 무시된다. n-그램에 기반을 둔 부분문자열 검색과 근사질의는 문자열 간의 편집거리(edit distance)에 기반을 둔다. 두 스트링 $o1$과 $o2$가 서로 편집 거리 k 안에 존재한다고 할 때 이 두 스트링의 n-그램들이 서로 일치할 것이고, 일치하지 않는 n-그램의 수는 적어도 k개를 넘지 않는다. 만약 k개가 넘는다고 하면, 두 스트링은 서로 공유하는 부분 문자열이 없다고 간주한다. 즉, 질의 문자열 o_q에 대해 $(o_s, o_q) \le k$이면, 두 문자열에서 서로 일치하는 n-그램의 수는 $\ge |o_q| - (k+1)n+1$이 된다. 질의 문자열과 DB에 저장된 문자열 o과의 비교는 2단계로 이루어진다. 첫 번째 단계는 각 질의 용어에 대한 n-그램들을 작성하는 단계이고, 두 번째 단계에서는 이들 n-그램을 가지고, DB에 저장된 n-그램들과 동등 검사를 통해 각 n-그램이 추출된 용어들을 검색한다.

본 시스템에서는 용어 기반 역인덱스를 검색의 기본 구조로 이용하고 있으므로, 가중치도 용어에 대해서 부여된다. 이에 따라 GetWeatherInfoByZipcode와 같은 연산자 이름은 하나의 용어로써 인식되고 이에 대한



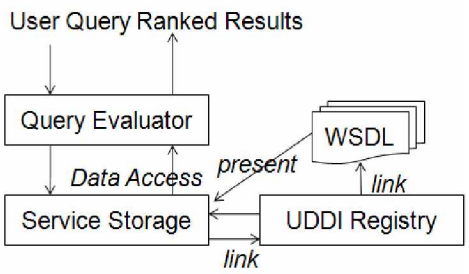
〈그림 4〉 부분문자열을 포함하는 경우들의 예

가중치가 계산되며, 질의어에 대한 가중치는 질의어가 포함된 용어의 가중치로 계산한다. 용어가 부분문자열을 포함하는 경우의 예는 <그림 4>에서 보이는 바와 같이 크게 4가지 유형이 존재할 수 있다.

첫 번째의 경우는 질의어 t_k에 대하여 질의어를 포함하는 t_i의 가중치를 그대로 t_k의 가중치로 한다. 두 번째의 경우는 두개의 질의어 t_k와 t_{k+1}에 대해 t_i의 가중치를 각각 t_k와 t_k+1의 가중치로 한다. 세 번째의 경우는 두 번째 경우와 동일하지만 t_k와 t_k+1의 n-그램들 중에서 일부가 동일한 경우이다. 이 경우에는 부분문자열 일치의 조건에 부합하지 않는 용어가 질의어의 n 그램과 일치하는 n 그램의 수가 $\geq |\sigma_q| - (k+1)n+1$일 수 있다. 이에 따라 본 논문에서는 n-그램의 시작 위치 값을 n-그램과 같이 기록하고, 이를 검색에 이용한다. 즉 질의어의 n-그램의 첫 번째 위치와 용어에서 해당 n 그램과 일치하는 n 그램의 위치의 차이는 언제나 일정하다는 조건을 부여함으로써 이를 해결한다. 이의 구현은 제 4장에서 보인다. 마지막은 질의어 t_k가 한 용어 t_i에서 반복적으로 출현하는 경우이다. 이 경우에는 원래 t_i의 가중치에 t_k가 출현하는 횟수만큼 곱한 값을 가중치로 한다.

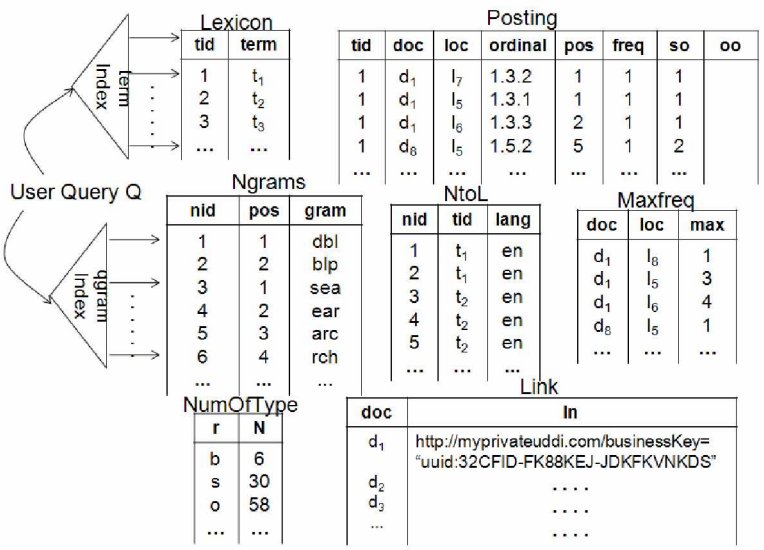
4. 시스템의 구현

<그림 5>는 본 논문에서의 서비스 검색 기법을 지원하기 위한 시스템 구조를 보인다. 해당 시스템은 관계형 DBMS를 이용하여 구현되었으며, 웹 서비스 레지스트리인 UDDI와의 연계를 수행한다. 서비스 설명정보의 저장소는 DB 테이블들과 이들에 결쳐지는 인덱스들로 구성된다. DB 구축을 위해 먼저 UDDI의 programmer's API를 호출하여 businessEntity 정보를 가져온다. 또한, UDDI에서 링크로 연결되는 WSDL 문서 또한 가져와 <그림 1>의 구조를 가지는 간략한 서비스 설명정보를 생성한다. <표 1>은 간략한 서비스 설명정보의 생성을 위한 알고리즘을 보인다. 이후 이 문서를 파싱하여 DB에 저장한다.



〈그림 5〉 시스템 구조

<그림 6>은 본 논문에서의 서비스 설명정보 저장을 위한 DB 테이블의 구조를 보인다. 본 논문에서는 단어 기반 역인덱스를 DB 테이블로 구축한 후, 각 단어에 대한 n-그램들을 별도의 테이블로 구성하고, 이들 n-그램들과 용어 테이블 간의 동등-조인을 통해 부분 문자열 검색을 지원하였다. 이는 n 그램 자체를 용어로 인식하고 n 그램을 기준으로 역인덱스를 구축하고 가중치를 부여하는 [5]의 연구와는 다르게 n-그램에서 n의 크기는 용어 가중치의 계산에는 영향을 미치지 않는다. 다만 검색 문자열의 최소 길이는 $|n|$으로 제한된다.



〈그림 6〉 서비스 설명정보 저장을 위한 테이블 구조

역 인덱스는 Lexicon과 Posting 테이블로 구현이 되며, 공백 문자로 구별되는 각 용어들은 Lexicon 테이블에 저장되고, 각기 유일한 용어 식별자, tid를 부여받는다. 그리고 빠른 스캔을 위해 Lexicon.term 컬럼에 대해 B' 트리 인덱스를 구축한다. Posting 테이블은 해당 용어의 빈도와 물리적, 개념적 위치 정보를 저장한다. Posting 테이블에서 doc 칼럼은 문서의 ID를 나타내며, loc, ordinal은 엘리먼트의 경로 id, ordinal은 뷰어 넘버링 기법의 일종인 DLN[3]으로 표시되는 각 노드들의 트리 상에서의 순서 정보를 나타낸다. pos와 freq는 각각 한 용어가 해당 엘리먼트의 텍스트에서 출현하는 오프셋과 빈도($freq_{ij}(d_h)$)를 기록한다. 가령 1번째 열의 경우 tid 1인 용어는 PathId 7인 엘리먼트의 텍스트에서 첫 번째 용어(pos = 1)로 출현하며, 한번만(freq = 1) 출현함을 의미한다. so와 oo 칼럼은 IEF^{r}_{ij}의 계산을 위해 이용된다. n^{r}_{ij}의 값은 질의의 r값에 의해 변경될 수 있기 때문에, 우리는 Posting 테이블의 열들을 r에 따라 그룹화할 필요가 있으며 이를 위해 so와 oo 칼럼 값을 이용한다. so 칼럼은 <그림 3>에서 보인 개념적인 트리 상에서 각기 구별되는 서비스들에 대한 식별 번호이며, oo 칼럼은 연산자들에 대한 식별 번호이다. 만약 so와 oo 칼럼이 각각 1이면, 이는 첫 번째 서비스의 첫 번째 연산자임을 나타낸다. 비즈니스의 경우는 각 서비스 설명정보 문서의 식별자인 doc 칼럼 값을 그대로 이용한다. 역 인덱스로의 동작은 다음과 같이 두 테이블 간의 동등 조인을 통해 수행된다.

```
select L.term, P.doc, P.loc, P.ordinal
from lexicon as L, posting as P
where L.tid = P.tid and L.term = 't_i'
and P.loc = l_i [or L.term = 't_i + 1' and
P.loc = l_i + 1] group by P.doc
```

〈표 1〉 UDDI와 WSDL 정보로부터 간략한 서비스 설명 정보 생성

```
알고리즘 1. UDDI와 WSDL 정보로부터 간략한 서비스 설명정보 생성
입력 : UDDI의 Programmer's API 인터넷 주소
출력 : 간략한 서비스 설명정보 wsdesc
변수 : node_ptr ← null, ptr ← null

find_business(%) 메시지를 UDDI의 Programmer's API에 전송하고 해당 결과 businessList 메시지를 획득;
ptr ← businessList의 첫 businessInfo에 대한 포인터;
WHILE ptr ≠ null
      key ← businessInfo의 businessKey;
      getBusinessDetail()을 해당 key 값으로 호출하고 해당 결과를 businessDetail 메시지로 얻음;
      businessDetail 메시지를 SAXParser로 파싱;
      IF StartDocument() SAX 이벤트 발생 THEN
          XML 문서 wsdesc 생성;
      ELSEIF StartElement() SAX 이벤트 발생 THEN
          CASE 엘리먼트 이름이 OF
              "businessEntity" :   wsdesc의 루트를 "businessEntity"로 설정;
                                   node_ptr ← the root, wsdesc 의 businessEntity; BREAK
              "name" :
              "description" :      해당 엘리먼트를 복사하여 node_ptr의 자식 노드로 삽입; BREAK
              "businessService" : businessService 엘리먼트를 생성하여 node_ptr의 자식 노드에 삽입;
                                   node_ptr ← 새로 삽입된 busienssService; BREAK
              "accessPoint" : IF 애트리뷰트 값이 "wsdlDeployment"이면 THEN
                              CALL read_WSDL_from_URI 프로시저를 이 엘리먼트의 텍스트를 주고 호출
                              ELSE do nothing;
                              ENDIF BREAK
          ENDCASE
      ELSEIF EndElement() SAX 이벤트 발생 THEN
          IF 엘리먼트 이름이 "businessService" THEN
              node_ptr ← 루트, 즉 wsdesc의 businessEntity;
          ELSEIF 엘리먼트 이름이 "businessEntity" THEN
              node_ptr ← null;
          ENDIF
      ENDIF
      ELSEIF EndDocument() SAX 이벤트 발생 THEN
          ptr ← businessList의 다음 businessInfo의 ptr;
      ENDIF
ENDWHILE
```

추가하여, Link 테이블은 해당 서비스 설명정보를 가져온 UDDI에 대한 REST 인터페이스의 주소를 기록하며, NumOfType과 Maxfreq 테이블은 각각 용어 가중치의 계산에 있어 N^T과 $maxfreq_j(d_h)$ 값의 기록을 위해 이용된다. 이 예에서 비즈니스의 수(r b)는 6이며, 문서 1의 6번째 경로에서 가장 빈도가 높은 용어의 수는 4이다.

〈표 2〉 용어에 대한 가중치의 계산

```
알고리즘 2. 용어에 대한 가중치의 계산
입력 : 용어 ti, 위치 값 lj, 루트 r
출력 : r이 루트인 서브 트리에서 j 위치의 용어 ti 의 가중치, whij^r
변수 : int NumOfType; //r의 유형에 따른 서브 트리의 수
       int nid; //n 그램에 대한 ID
       String stmt;

NumOfType ← executeSQL ("select NO.N from NumOfType NO where NO.r = '$r' ");

stmt ← "select L.term, p.doc, p.loc, (p.freq/M.max) * (log(NumOfType/count_rooted($ti, $lj, $r))) + 1 as
        Weight
        from Lexicon L, Posting P, MaxFreq M
        where L.tid = P.tid and P.doc = M.doc and P.loc = M.loc and
        L.term = '$ti' and P.loc = "$lj"
        group by P.doc";
RETURN executeSQL(stmt);


FUNCTION count_rooted()
입력 : ti, lj, r
출력 : nijr;
변수 : String stmt //a SQL statement
IF r = b
THEN stmt ← "select count(P.doc) from Lexicon L, Posting P
             where L.tid = P.tid and L.term = $ti and P.loc = $lj group by P.doc";
ELSE IF r = s
THEN stmt ← "select count(P.doc, P.so) from Lexicon L, Posting P
             where L.tid = P.tid and L.term = $ti and P.loc = $lj group by P.doc, so";
ELSE IF r = o
THEN stmt ← "select count(P.doc, P.so, P.oo) from Lexicon L, Posting P
             where L.tid = P.tid and L.term = ti and P.loc = lj group by P.doc, so, oo";
ENDIF

RETURN executeSQL(stmt);
```

<표 2>는 시스템에서 이 테이블들을 이용하여 용어에 대한 가중치를 SQL 질의로 처리하는 방법을 보인다.

부분 문자열 검색을 위해서 앞서 언급한 바와 같이 n-그램을 이용한다. n-그램의 저장을 위해 Ngrams라는 별도의 테이블을 유지하며, n-그램과 용어와의 관계성은 NtoL 테이블을 통해 M : N 대응 관계를 명시한다. <표 3>은 서비스 설명정보에 포함된 용어 t_i에 대해 n-그램들을 생성하고 테이블에 저장하는 절차를 설명한다.

〈표 3〉 용어에 대한 n-그램의 생성과 저장

```
알고리즘 3. 용어에 대한 n-그램의 생성과 저장
입력 : (용어 ti와 대응하는 tid)의 쌍; int n; //n-그램의 크기
변수 : String text; //부분 문자열을 저장하기 위한 공간
   int nid; //n 그램에 대한 ID

FOR i = 0에서 i < length(str) - n + 1일 때까지
     text = substring(str, i, i+n);
     IF 해당 텍스트가 i 위치에 있는 n-그램이 이미 존재하는 경우
     THEN nid ← 해당 텍스트에 대한 n-그램 id;
     ELSE Ngrams 테이블에 pos ← i, gram ← text 값으로 삽입;
          nid ← 새로 삽입한 n-그램에 대한 n-그램 id;
     ENDIF
     IF (nid, tid) 연이 NtoL 테이블에 이미 존재하는가
     THEN do nothing;
     ELSE NtoL 테이블의 tid 칼럼 ← tid 값으로 삽입;
     ENDIF
ENDFOR
```

n-그램을 이용한 부분문자열 검색은 간단하게 SQL 질의를 통해 수행될 수 있다. <그림 6>은 n-그램을 이용한 부분문자열 검색에 대한 도해이다. 예를 들어 연산자의 이름이 "GetWeatherInfoByZipcode"이고, 사용자의 질의는 "weather"라고 하자. 먼저 사용자의 질의어에서 각 용어에 대해 n-그램을 생성하고, 이를 임시 테이블 TN에 저장한다. <그림 7>에서보이는 바와 같이 $|q|$ = 7인 용어에 대해 5개의 튜플을 가지는 임시 테이블 TN를 생성한다. 또한 연산자의 이름에서 추출된 n-그램들은 Ngrams 테이블에 저장되어 있다. 이 때 질의어에 대한 부분 문자열 검색은 이 Ngrams와 TN 테이블 간의 조인을 통해 이루어진다. 만약 유사 질의가 아닌 단순 부분 문자열 일치 검색의 경우 편집 거리 $k = 0$이며, 해당 TN 테이블의 튜플들과 일치하는 n-그램들을 추출하여, 용어별로 그룹화하고 그 수를 계산함으로써, 해당 용어를 부분 문자열로 포함하는지를 검사한다.

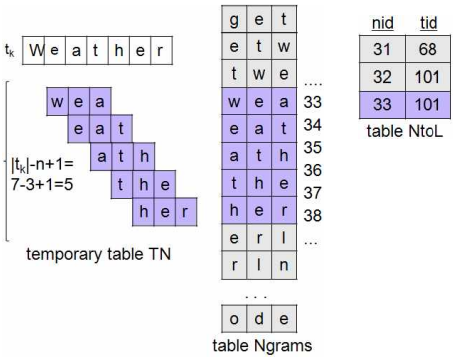
5. 실험 및 평가

지금까지 서비스 발견에 대한 평가를 위해 고안되거나 이용된 표준화된 평가 도구가 없기 때문에, 본 논문의 서비스 검색 기법을 관련 연구들과 직접적으로 완전하게 비교하기에는 어려움이 있다. 이에 우리는 본 논문의 서비스 검색 기법을 평가하기 위해 WWW에서 WSDL 파일들을 구한 후, UDDI에 해당 서비스들에 대한 추가적인 설명과 함께 이들을 입력하였다. 시스템은 MySQL을 DBMS로 하고, JDBC 2.0 API를 이용하여 개발되었다. <표 5>는 본 시스템의 평가를 위해 사용된 실험 데이터의 통계 정보를 보인다.

〈표 4〉 용어를 이용한 부분 문자열을 포함하는 용어의 탐색

```
알고리즘 4. n 그램을 이용한 부분 문자열을 포함하는 용어의 탐색
입력 : 질의 키워드 tk, n-그램의 크기 n
입력전제조건 : n은 알고리즘 3의 n과 동일해야 함.
출력 : 용어 ti에 대한 id list tid[max]
변수 : int diff[max]; //첫번째 gram은 위치 값을 기록하기 위한 배열

tk에 대한 n-그램의 생성
생성한 n-그램을 임시 테이블 TN(gram, pos)으로 저장 // pos는 원 문자열에서 해당 gram이 시작하는 위치
 // TN의 첫 n gram과 일치하는 Ngrams의 n gram과의 위치 차이를 구하기
diff[] ← executeSQL("select Ngrams.pos TN.pos from Ngrams, TN
                    where and Ngrams.gram  TN.gram and TN.pos  1");
i = 0;
WHILE diff[i] ≠ NULL
      tid[i] ← executeSQL("select NtoL.tid from Ngrams, NtoL, TN
                where Ngrams.nid = NtoL.nid and Ngrams.gram = TN.gram
                and Ngrams.pos TN.pos| = $diff[i] group by NtoL.tid
                having count(*) >  length(tk) - (k+1)n + 1");
    i++;
ENDWHILE
RETURN tid[]
```



〈그림 7〉 n-그램을 이용한 부분문자열을 포함하는 용어의 탐색

〈표 5〉 실험 데이터의 통계량

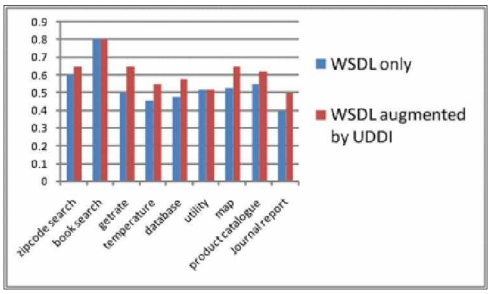
비즈니스의 수	서비스의 수	인산자의 수
50	141	968

UDDI에서 비즈니스의 수 : 50
WSDL 파일들의 전체 크기 : 2.14 MB

<그림 8>은 각 질의에 따른 정확률(precision)을 보인다. 그래프에서 보인 바와 같이 총 9개의 키워드 질의를 가지고 실험을 수행하였으며, 이들은 root(q)의 값에 따라 3개의 그룹(연산자, 서비스, 비즈니스)로 분류된다. 첫 번째 질의 그룹은 질의 키워드 zip code search, book search, getrate를 가지고 연산자를 찾기 위한 질의이다. 이 때 질의 결과는 operation 엘리먼트를 루트로 하는 서브트리들로, 각기 다른 operation 엘리먼트들 간의 식별은 이전에 설명한 바와 같이 노드마다 유일하게 구별되는 듀이 넘버링 값을 통해 이루어진다. 두 번째 그룹은 질의 키워드 temperature, database, utility와 관련한 서비스를 찾기 위한 질의로, 질의 결과는 businessService를 루트로 하는 서브 트리가 된다. 마지막 그룹은 business를 루트로 하는

시보 트리를 검색 결과로 하며, map, product catalog, journal report를 키워드로 한다.

그림에서 보는 바와 같이 WSDL 정의만을 서비스 검색에 활용하는 경우보다 UDDI에 기록된 UDDI 데이터 또한 검색에 활용하는 경우 정확률이 모든 질의의 경우에 대해 대체적으로 증가함을 확인할 수 있다. 이는 UDDI에서 추출된, 비즈니스와 서비스와 관련한 4개의 경로(<그림 1>에서 PathId 2, 3, 5, 6)상의 노드의 용어들을 가지고 서비스에 대한 설명정보가 확장되었기 때문이다.

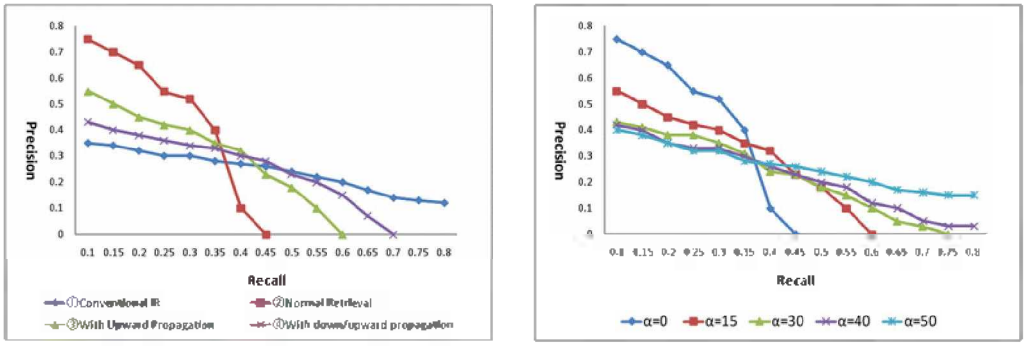


〈그림 8〉 각 질의에 따른 정확률의 비교

<그림 9>는 본 논문의 서비스 유사도 계산 기법을 9개의 질의에 대하여 각기 다른 α, β 계수를 두면서 정확률과 재현율을 구한 후 이를 평균화한 결과를 보인다.

왼쪽의 그림에서 1번째 그래프는 평문을 대상으로 하는 TF*IDF의 용어 가중치를 가지는 정보 검색을 통해 구해진 정확률 재현율 그래프이다. 전체적으로 낮은 정확률을 보이지만, 재현율의 범위는 넓은 것을 알 수 있다. 이는 구조화된 XML 문서의 특징을 무시하고, 단순히 하나의 서비스 설명정보를 하나의 문헌으로 간주함에 따라 위치에 따른 용어의 빈도와 역문헌빈도만으로 유사도를 계산하였기 때문이다.

2번째 그래프는 정의 4의 유사도 계산 방법에 따른 그래프이다. 1번째 그래프와 비교해 볼 때 낮은 재현율 범위에서 평문을 대상으로 하는 정보 검색 기법보다 높은 정확률을 보인다. 이는 위치에 따른 용어들의 빈도 계산을 통해 보다 정확도가 보다 향상되었기 때문이다. 하지만, 이에 따라 재현율은 상대적으로 낮아지게 된다. 왜냐하면 동일한 용어가 다른 위치에 존재하는 경우에는 검색이 되지 않기 때문이다. 때문에, 반대로 높은 재현율 범위에서는 낮은 정확도를 보이게 된



〈그림 9〉 본 논문의 알고리즘에 대한 재현율-정확률 그래프

다. 3, 4번째 그래프는 이의 보간을 위해 확장된 정의 5의 식에 따라 하위 개념과 상위 개념에 따라 측정된 유사도를 전파시킨 경우를 보인다. 3번째 그래프는 $\alpha = 15$,$\beta = 0$인 경우이며, 4번째 그래프는 $\alpha = 15$, $\beta = 15$인 경우를 보인다. 결과적으로 정의 5의 식을 통해서 평문을 대상으로 한 정보 검색 기법보다는 낮은 재현율에서 정확률이 높고 정의 4의 식보다 재현율이 우수한 실험 결과를 얻을 수 있음을 확인할 수 있다.

<그림 8>은 α값의 변경에 따른 재현율-정확률의 변화를 보인다($\beta = 0$). 그래프에서 보이는 바와 같이 하위 개념의 유사도를 보다 많이 전파하는 경우 정확률은 떨어지지만 재현율은 증가하는 결과를 얻을 수 있었다. 하지만, $\alpha = 30$이상 부터는 아래 개념의 상위로의 유사도 전파가 검색 성능의 향상에 더 이상 도움을 주지 못함을 확인할 수 있었다.

6. 결론 및 향후 연구 방향

본 논문에서는 웹 서비스 검색을 위한 실용적이고, 효과적인 검색 기법에 대해 소개하였다. 본 논문에서는 서비스 발견 문제를 서브 트리가 임의로 결정될 수 있는 특정 구조의 XML 트리를 대상으로 하는 정보 검색 문제로 간주하고, 이의 해결 방법을 모색해 보았다. 이를 위해 기존의 정보 검색 기법을 특정 XML 구조에 적합하도록 수정하였고, DBMS의 기능을 활용한 서비스 유사도의 검색 방법을 고안하였다.

향후 연구로는 서비스 발견 기법들의 평가를 위한 표준화된 평가 도구의 개발과 보다 본 기법의 보다 다양한 조건에서의 실험, 평가가 요구된다. 또한, 사용자의 작업량을 최소화할 수 있는 서비스의 검색과 연산자의 파라미터인 XML 메시지의 형식 또한 고려할 수 있는 검색 기법의 개발이 요구된다.

참 고 문 헌

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

[2] Berners Lee, T., Hendler, J., and Lassila, O., "The Semantic Web, Scientific American," Vol. 284, No. 5, 2001, pp. 34-43.

[3] Böhme, T. and Rahm, E., "Supporting Efficient Streaming and Insertion of XML Data in RDBMS," Proceedings of DIWeb(Workshop on Data Integration over the Web), 2004, pp. 70-81.

[4] Bray, T., Paoli, J., and Sperberg-McQueen, C. M., "Extensible Markup Language(XML) 1.0," W3C Recommendation 10 February 1998, http://www.w3.org/TR/1998/REC xml 19980210, 1998.

[5] Cavnar W. B., "Using An N-Gram-Based Document Representation with A Vector Processing Retrieval Model," Overview of the Third Text Retrieval

Conference (Trec 3)), NIST special publication, Vol. 500225, 1995, pp. 269 277.

[6] Cohen, S., et al., "XSearch : A Semantic Search Engine for XML," Proceedings of VLDB, Vol. 29, 2003, pp. 45 56.

[7] Curbera, F., et al., "Unraveling the Web services web : An introduction to SOAP," WSDL, and UDDI, IEEE Internet Computing, Vol. 6, No. 2, 2002, pp. 86-94.

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

[9] Guo, L., et al., "XRank : Ranked Keyword Search over XML Documents," Proceedings of ACM SIGMOD, 2003, pp. 16 27.

[10] Gravano, L., et al., "Approximate string joins in a database (almost) for free," Proceedings of VLDB(Very Large Data Bases), 2001, pp. 491-500.

[11] Kokash, N., A Comparison of Web Service Interface Similarity Measures, University of Trento, Technical Report : DIT 06 025, 2006.

[12] Lee, D., Chuang, H., and Seamons, K., "Document ranking and the vector-space model," IEEE Software, Vol. 14, No. 2, 1997, pp. 67 75.

[13] Lee, J. and Ahn, J., "Using n Grams for Korean Text Retrieval," Proceedings of ACM SIGIR, 1996, pp. 216-224.

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

[15] Navarro, G., Baeza-Yates, R. A., Sutinen, E., and Tarhio, J., "Indexing Methods for Approximate String Matching," IEEE Data Engineering Bulletin, Vol. 24, No. 4, 2001, pp. 19 27.

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

[17] Platzer, C. and Dustdar, S., "A Vector Space Search Engine for Web Services," Proceedings of ECOW(IEEE European Conference on Web Services), 2005, pp. 62-71.

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

[19] Sajjanhar, A., Hou, J., and Zhang, Y., "Algorithm for Web Services matching," Proceedings of APWeb, Springer, 2004, pp. 665-670.

[20] Srivastava, B. and Koehler, J., "Web Service Composition-Current Solutions and Open Problems," Proceedings of ICAPS 2003 Workshop on Planning for Web Services http://citeseer.ist.psu.edu/srivastava03web.html, 2003.

[21] Wang, Y. and Stroulia, E., "Semantic Structure Matching for Assessing Web Service Similarity," Proceedings of ICSOC (International Conference on Service-Oriented Computing), LNCS 2910, 2003, pp. 194-207.

[22] Xu, Y. and Papakonstantinou, Y., "Efficient keyword search for smallest LCAs in XML databases," Proceedings of SIGMOD, 2005, pp. 527-538.

[23] Zhuang, Z., Mitra, P., and Jaiswal, A., "Corpus based Web Services Matchmaking, Workshop on Exploring Planning and Scheduling for Web Services," Grid and Autonimic Computing, 2005, pp. 46-52.

[24] Zhuge, K. and Liu, J., "Flexible retrieval of Web services," The Journal of Systems and Software, Vol. 70, No. 1-2, 2004, pp. 107-116.

저 자 소 개

이성하 (E-mail : bart7449@gmail.com)
1998년 충남대학교 공과대학 정보통신공학과 (학사)
2000년 충남대학교 공과대학 정보통신공학과 (석사)
2006년 충남대학교 공과대학 컴퓨터공학과 (박사)
2006년 한국전자통신연구원 디지털홈연구단 인터넷서비스그룹 postdoc.
2007년~현재 충남대학교 소프트웨어연구소 전임연구원
관심분야 데이터베이스, XML, 웹 서비스

이규철 (E-mail : kclee@cnu.ac.kr)
1984년 서울대학교 공과대학 컴퓨터공학과 (학사)
1986년 서울대학교 공과대학 컴퓨터공학과 (석사)
1990년 서울대학교 공과대학 컴퓨터공학과 (박사)
1994년 미국 IBM Almaden Research Center 초빙 연구원
1995년~1996년 미국 Syracuse University 초빙 교수
2001년~현재 전자상거래 표준화 통합 포럼 전자거래 기반 기술위원회 위원장
2003년~현재 한국전자거래학회 편집이사
2003년~현재 웹 코리아 포럼 부위원장
현재 충남대학교 공과대학 컴퓨터공학과 교수
관심분야 데이터베이스, XML, 웹 서비스, 시멘틱 웹 서비스, 유비쿼터스 컴퓨팅

김경옥 (E-mail : kokim@etri.re.kr)
1976년 서울대학교 (학사)
1987년 미국 오하이오 주립대학 (전산학 석사)
1998년 충남대학교 컴퓨터공학과 (박사)
1981년~1984년 미국 오하이오 주립대학 기술원
1984년~1988년 미국 오하이오 주립대학 연구조교
1988년~1997년 시스템공학연구소 선임연구원
1998년~현재 한국전자통신연구원 텔레매틱스연구그룹 공간정보연구팀장
관심분야 멀티센서(위성, 항공영상, 레이다, SAR, 현장센서등) 데이터를 이용한 분석 및 3차원 건물, 지형 등을 추출, 3차원 모니터링 및 맞춤형 국토정보 제공기술 웹 서비스 시스템 구축