통합 개념 모델에 기반한 시맨틱 웹 서비스 탐색

두희준*, 신동훈**, 이경호***

요 약
시맨틱 웹 서비스 탐색은 응용프로그램의 개념으로 기술된 서비스와 사용자 요구 사항 간에 개념 간 매칭을 통해 서비스를 탐색하는 방법이다. 그러나 응용프로그램의 개념은 해석하는 방법에 따라 서로 다른 의미로 이해될 수 있다. 기존 연구는 제한된 수준의 개념 해석 방식을 사용하고, 명확한 개념 기술 방법을 제공하지 않기 때문에 매칭 오류가 발생할 수 있다. 본 논문에서는 이러한 문제를 해결하기 위해 통합 개념 모델을 이를 적용한 웹 서비스 탐색 방법을 제안한다. 제안된 통합 개념 모델은 개념 탐색, 클레스, 프로퍼티를 이용하여 기존 방법 보다 유연하고 명확한 개념 기술 방법을 제공한다. 또한 본 논문은 통합 개념 모델에 기반하여 기술된 서비스와 사용자 요구 사항 간에 매칭하는 방법을 제안한다. 이에 다대다 복합 매칭을 지원하여 보다 정교한 서비스 탐색이 가능하다. 다양한 형태의 사용자 요구 사항에 대해 실험한 결과, 제안된 방법은 기존 연구보다 더 높은 정확률과 재현율을 보였다.

Search for Semantic Web Services Based on the Integrated Concept Model

Hwa-Jun Du*, Dong-Hoon Shin**, Kyong-Ho Lee***

ABSTRACT
Semantic Web Services Discovery matches between users’ requirements and the ontological description of Web Services. However, concepts of an ontology can be interpreted differently according to a point of view. Previous works are limited in interpreting concepts. Since they lack a precise scheme of describing the advertisments and requirements of services and users, respectively, and even do not support a sophisticated matching, semantic mismatches may occur. This paper presents a sophisticated method of discovering Web services. The proposed method facilitates specifying semantics precisely and flexibly based on a proposed integrated concept model. Additionally, more sophisticated discovery is supported by computing complex matchings with many-to-many relationships. Experimental results show that the proposed method performs more efficiently for various kinds of user requests, compared with previous works.

Key words: Web Services(웹 서비스), Discovery(탐색), OWL-S, Integrated Concept Model(통합 개념 모델), Complex Matching(복합 매칭)

*교신저자(Corresponding Author): 이경호, 주소: 서울시 서대문구 신촌동 134 연세대학교 공과대학 제3공학관 527-1호, 인터넷컴퓨팅연구실(120-749), 전화: 02)2123-5712, FAX: 02)365-2579, E-mail: khlee@cs.yonsei.ac.kr
**정회원, 연세대학교 컴퓨터과학과
***홍심원, 연세대학교 컴퓨터과학과
※이 논문은 서울시 산학연 협력사업(10705)의 지원에 의해 연구되었음.
1. 서론

웹 서비스 (Web services)는 플랫폼에 상관없이 어플리케이션 간의 상호 운용을 지원하는 소프트웨어 기술로서, B2B 통합과 전자상거래를 위한 핵심 기술로 인정되고 있다. 웹 서비스는 SOAP[1], WSDL (Web Services Description Language)[2], UDDI (Universal Description Discovery and Integration)[3]와 같은 표준을 사용하여 서비스의 제작 및 서비스 간의 상호 작용을 지원한다. 최근 들어 웹 서비스가 널리 사용되고 이용 가능한 웹 서비스의 수가 증가함에 따라 원하는 기능의 웹 서비스를 정확하게 탐색하는 것이 중요한 이슈가 되고 있다.

현재 웹 서비스 레이스터리 표준인 UDDI는 기위드 기반의 단순한 탐색 방법을 제공한다. 이에 보다 정확한 수준의 탐색을 지원하기 위한 시멘틱 웹 서비스 탐색 방법[4-11]이 제안되었다. 시멘틱 웹 서비스 탐색 방법은 서비스의 기반 및 오, 출력 정보 등을 온톨로지를 이용하여 기술한 후, 이를 기반으로 탐색하기 때문에 기존 기위드 기반 탐색 방법에 비해 정확성이 높다.

시멘틱 웹 서비스 탐색은 웹 서비스와 사용자 요구 사항을 온톨로지의 개념과 매핑시켜 기술하고 이를 간의 의미적 매핑을 통해 서비스를 탐색한다. 온톨로지[12,13]는 그림 1(a)와 같이 특정 도메인의 개념 정보를 구조화한 것으로서 클래스(class)와 프로퍼티(property)를 통해 기술된다. 예를 들어, 자동차 정보를 제공하는 서비스의 출력은 그림 1(a)의 Car 클래스와 매핑되어 기술될 수 있다. 의미적 매핑은 온톨로지를 기반으로 기술된 서비스의 입력, 출력 개념과 서비스 제공자가 기술한 입력, 출력 개념 간에 이루어진다. 그림 1(b)는 사용자가 원하는 출력의 개념이 Car이고 서비스의 출력에 해당하는 개념 역시 Car인 경우, 매칭이 성립되는 것을 나타낸다. 그러나 사용자 요구 사항과 정확히 일치하는 서비스가 항상 존재하지는 않으므로 사용자의 요구사항과 유사한 서비스를 탐색해야할 필요가 있다. 따라서 개념 간의 유사도는 정교하게 계산되어야 하며 이를 위해 기존 연구는 개념 모델을 사용한다.

개념 모델은 온톨로지 개념에 대한 해석 방식으로서 개념 간 매칭 정도를 결정하는데 사용된다. 본 논문에서는 기존 연구에서 사용하는 개념 모델을 개념 간의 포함 관계에 대한 해석 방식에 따라 크게 범위 모델(coverage model)과 상속 모델(inheritance model)로 구분한다. 범위 모델은 온톨로지 상의 상위 클래스가 하위 클래스에 대한 요청을 만족한다고 해석한다[14]. 예를 들어, 그림 1(a)의 Vehicle를 제공하는 서비스는 Truck과 Car도 제공한다고 해석한다. 만약 사용자가 Car를 출력하는 서비스를 요청할 때 Vehicle를 출력하는 서비스가 존재한다면 해당 서비스를 매핑된 서비스로 간주한다. 한편 Vehicle를 제공하는 서비스는 Car를 제공할 수 있지만 Car의 하위 클래스인 Sedan과 SUV에 대한 제공은 보장하지 못한다고 해석한다. 한편 상속 모델은 객체 지향 모델과 유사하여 하위 클래스가 상위 클래스를 상속하

![Graph 1. Semantic Web Service Ontology of Car]
여 상위 클래스에 대한 요청을 만족한다고 해석한다 [11]. 예를 들어, 그림 1(a)에서 사용자가 Car를 출력 하는 서비스를 요청할 때 SUV를 출력하는 서비스가 존재한다면 사용자는 SUV 정보를 통해 보다 자세한 출력을 얻을 수 있도록 배치된 서비스로 간주한다.

그러나 기존 연구는 한 가지 개념 모델을 적용하 여 매핑을 기술한다는 측면에서 제한적이다. 온톨로 지 개념은 사용자에 따라 다양하게 해석될 수 있기 때문에 사용자가 보다 다른 개념 해석 방식을 지원 하는 유연한 개념 기술 방식이 필요하다. 또한 기존 연구는 클래스를 기준으로 사용자가 요청한 개념과 서비스가 제공하는 개념이 같으면 서로 매칭한다. 그러나 보다 정교한 매칭 위해서는 클래스뿐만 아니라 프로퍼티간의 매칭을 지원해야 한다. 따라서 본 논문에서는 기존의 개념 간 매핑 방식을 모두 지원하 며 프로퍼티 수준에서 개념 간 매핑을 기술할 수 있는 정교한 수준의 개념 모델을 통합 개념 모델을 제안한다.

한편 기존 개념은 개념 간의 일대일 매칭에 기반 한 단순 매칭을 통해 서비스를 찾는다. 그러나 단순 매칭만으로는 사용자 요구 사항을 만족하는 서비스 가 탐색되지 않을 수 있다. 예를 들어, 사용자가 요구 하는 단일 출력을 서비스의 출력 집합이 만족시키는 경우, 단순 매칭을 사용하는 기존 방법은 해당 서비스를 찾을 수 없다. 따라서 제안된 방법은 다수의 입력 출력 간에 일대일 매칭뿐만 아니라 다다다의 복합 매칭을 지원한다. 본 논문에서는 제안된 방법의 성능 을 평가하기 위해 서로 다른 두개의 도메인에 속하는 다수의 서비스를 대상으로 실험한 결과, 평균 95.6%의 정확률과 93.8%의 재현율을 나타내어 기존 연구 보다는 우수하였다.

본 논문의 구성은 다음과 같다. 2장에서는 시맨틱 웹 서비스 탐색 관련 기존 연구의 특징을 간략히 기술하고 제안된 통합 개념 모델을 소개한다. 3장에서는 제안된 시맨틱 웹 서비스 탐색 방법을 도메인 정보 획득, 서비스 및 사용자 요구 사항 기술, 서비스 출판 그리고 서비스 탐색으로 구분하여 자세히 기술 한다. 4장에서는 제안된 방법의 평가하기 위한 실험 결과를 기술한다. 끝으로 5장에서는 결론 및 향후 연구 방향에 대해 기술한다.

2. 관련 연구 및 제안된 통합 개념 모델

본 절에서는 기존의 시맨틱 웹 서비스 탐색 방법 의 특징 및 제안된 통합 개념 모델을 기술한다.

2.1 시맨틱 웹 서비스 탐색 방법

표 1은 기존 시맨틱 웹 서비스 탐색 방법의 특징을 간략히 기술하고 각각의 방법에서 사용된 개념 모델 을 보여준다.

<table>
<thead>
<tr>
<th>저자</th>
<th>연도</th>
<th>특</th>
<th>개념모델</th>
</tr>
</thead>
<tbody>
<tr>
<td>Srinivasan 등 [7]</td>
<td>2005</td>
<td>서비스 출판 시에 서비스의 매칭 정도를 사전에 저장하여 탐색 시 효율 높임, 탐색 시 범위 모델 기반으로 매칭</td>
<td>범위모델</td>
</tr>
<tr>
<td>Eldedawy 등 [8]</td>
<td>2004</td>
<td>사용자 요구 사항과 서비스 매칭 비 개념 친화 그래프와 개념 변환 그래프를 사용하여 개념 간 매칭 정도 평가</td>
<td>친화 관계 사용</td>
</tr>
<tr>
<td>Aggarwal 등 [9]</td>
<td>2004</td>
<td>개념 사이에 공통 부모, 형제, 부모 자식 관계를 이용한 매칭 방법 제안</td>
<td>상속모델</td>
</tr>
<tr>
<td>Akkiraju 등 [10]</td>
<td>2003</td>
<td>시맨틱 정보가 포함된 요청의 처리를 위해 UDDI 접의 API 확장, 후위 추론 알고리즘 기반의 자동화된 서비스 조합</td>
<td>범위모델</td>
</tr>
</tbody>
</table>


Elgedawy 등 [8] 은 온톨로지의 구성 요소를 개념, 기능, 역할, 그리고 규칙으로 표현하고 이들 기반으로 개념 간의 관계, 그래프(substitutability graph)와 변환 그래프(translation graph)를 생성한다. 서비스와 사용자 요구 사항은 온톨로지를 활용하여 서비스의 목적이, 선행 조건( precondition), 그리고 사후 조건( effect)을 통해 기능적 상호작용을 표현하는 Graph 모델을 사용하여 기술한다. 일반적인 개념 모델을 사용하지 않고 개념 사이의 관계와 변환 관계를 이용한다. 서비스 매칭 시에는 변환 그래프의 변환 그래프를 통해 두 개념이 상호간에 친화되고 변환될 수 있는지를 평가하여 유사도를 계산한다.

Aggarwal 등 [9] 은 에노테이션된 WSDL로 서비스를 기술하며 서비스 요청 시에는 도메인, 위치, 기능, 입력, 출력, 예외(exception), 선행조건, 사후조건 등을 기술한 서비스 템플릿을 생성하여 질의한다. 서비스 매칭 시에는 두 개념 간의 공통 분류, 형제, 부모 자식 관계를 이용하여 계산한다. 그러나 개념 간 공

t 동 부속 관계를 고려하기 때문에 서비스 탐색 시의 미적으로 완전히 연관이 없는 개념이 매칭되는 문제점이 있다.


Cardoso와 Sheth [11]는 웹 서비스 조합을 위해 작업 흐름에 필요한 서비스를 서비스 템플릿으로 기술하여 탐색한다. 탐색 시에는 사용자 유사도, 오�ToUpper이션 유사도, 그리고 서비스 유사도를 통해 적합한 서비스를 찾는다. 구문 유사도는 서비스 명 및 명세간의 유사도를 평가하고, 오�ToUpper이션 유사도는 QoS 정보의 만족도를 계산하여 서비스 유사도는 서비스의 입력저에 대하여 유사도를 계산한다. 서비스 유사도 계산 시 전술한 바와 같이 기존 연구는 서비스 출판 및 탐색 시에 단일의 개념 모델만을 지원한다는 측면에서 제한적이다. 예를 들어, 그림 2에서 서비스 1과 서비스 2를 범위 모델을 이용하여 기술하였으나 기술하지 않았으며 이는 서비스 1이 출력으로 Car 개념은 물론이고 Sedan과 SUV를 제공하며 서비스는 Vehicle는 물론이고 Truck과 Car를 제공한다. 따라서 사용자가 Car를 출력으로 하는 서비스를 요구하면 서비스 1과 서비스 2 모두 적합한 매칭 결과에 해당한다 [6,7,10]. 그러나 사용자가 Car를 Sedan과 SUV를 반드시 포함하는 개념으로 해석하여 요청한다면 서비스 2는 Sedan과 SUV의 제공을 보장할 수 없으므로 정확한 매칭이라 보기 어렵다. 한편 상속 모델은 상위 개념이 하위 개념의 일반화 개념임에 하위 개념을 합친 합집합을 제공하는 것이 불가능하다.
[5, 9, 11]. 예를 들어, 서비스 제공자가 그림 1(a)의 Truck과 Car를 하나로 합칠 개념을 단일 개념으로 제공하고자 한다면 상속 모델에서는 이를 기술할 수 없다.

또한 기존 연구는 개념을 클래스로 기술하여 사용자가 요청한 개념과 서비스의 개념이 같으면 매칭으로 간주한다. 그나 사용자가 요청한 개념과 서비스의 개념이 같지라도 사용자가 요구하는 프로퍼티가 실제 서비스가 제공하는 프로퍼티가 동일하지 않은 경우, 정확한 매칭을 보이 어려다. 예를 들어, 그림 3에서 사용자가 프로퍼티 name과 size를 포함하는 Car를 요구할 경우, 기존 방법은 프로퍼티를 기술하지 않으므로 Car만을 기술하게 되며 매칭 시에는 Car를 출력으로 제공하는 서비스를 찾는다. 그러나 그림 3에서 기존 방법을 통해 찾아진 서비스 중에서 name과 size를 제공하는 서비스는 없으므로 정확한 매칭이 보이 어려다.


2.2 통합 개념 모델

본 절에서는 기존 개념 모델의 문제점을 개선하기 위해 제안된 통합 개념 모델을 기술한다. 기존 개념 모델은 개념 기술 시 클래스만을 사용하고 개념의 범위를 명확하게 표현하지 않기 때문에 개념 간 매칭 오류가 발생할 수 있다. 제안된 방법은 해석 방식에 따라 달라지는 개념의 범위를 표현하기 위해 개념 탐색을 정의하고, 개념 기술의 불명확성을 개선하기 위해 클래스와 프로퍼티를 모두 활용하여 개념을 기술한다. 따라서 개념의 의미를 명확하게 기술할 수 있으며 탐색 시 정확성을 향상시킨다. 제안된 개념 탐색의 종류와 의미는 표 2와 같다.

Restriction 탐색은 상속 모델의 개념 기술 방식으로서 개념을 클래스 자체로 제한한다. Restriction 탐색은 사용자가 개념 기술 시 해당 개념 자체를 기술할 때 사용될 수 있다. Partial 탐색은 개념의 프로퍼티 중 일부를 이용하여 정보를 기술할 때 사용될 수 있다. Coverage 탐색은 범위 모델의 개념 기술

<table>
<thead>
<tr>
<th>개념 탐색</th>
<th>의미</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Restriction 탐색</strong></td>
<td>- 개념을 클래스 자체로 제한함 &lt;br&gt; - 해당 클래스에 속하는 모든 프로퍼티를 기술</td>
</tr>
<tr>
<td><strong>Partial 탐색</strong></td>
<td>- 클래스에 속하는 프로퍼티를 부분적으로 제공하는 개념을 기술 &lt;br&gt; - 클래스에 속하는 프로퍼티 중 일부 프로퍼티로 기술</td>
</tr>
<tr>
<td><strong>Coverage 탐색</strong></td>
<td>- 해당 개념의 자식 개념을 모두 포함하는 개념 탐색 &lt;br&gt; - 하위 개념들이 제공하는 모든 프로퍼티 포함</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(a) 클래스 Car의 프로퍼티 예.</th>
<th>(b) 부정확한 매칭의 예.</th>
</tr>
</thead>
<tbody>
<tr>
<td>사용자 요구사항</td>
<td>Car</td>
</tr>
<tr>
<td>name</td>
<td>size</td>
</tr>
<tr>
<td>displacement</td>
<td>color</td>
</tr>
</tbody>
</table>

그림 3. 기존 방법 적용 시 발생하는 부정확한 매칭의 예
방식으로서 하위 개념을 합친 개념을 기술하는 경우에 사용될 수 있다. 제안된 개념 모델은 먼저 해당 개념의 클래스를 기술하고, 해당 클래스의 개념 태입을 기술하며, 개념 태입에 맞는 프로퍼티 집합을 기술한다. 개념 모델의 프로퍼티 집합은 개념 태입에 따라 다른 프로퍼티 집합을 가지기 때문에 프로퍼티를 통해 해당 개념의 특성을 나타내게 된다. 제안된 통합 개념 모델은 다음과 같은 형태로 정의된다.

통합 개념 모델 = (클래스 명, Restriction | Partial | Coverage, 프로퍼티 집합).

그림 4는 통합 개념 모델을 기반으로 클레스 Agency를 개념 태입에 따라 다르게 기술한 예이다. 예를 들어, ①은 Agency 자체를 나타내는 개념이고, ②는 Agency 정보 중 이름과 email 정보만을 포함하는 부분화된 개념이며, ③은 AirAgency, CarAgency, 그리고 HotelAgency를 모두 포함하는 Agency를 나타내는 개념이다. 제안된 방법은 하나의 개념이 가질 수 있는 다양한 태임을 정의함으로써 기존의 개념 모델보다 보다 유연한 개념 해석이 가능하며 명시적으로 개념 태임을 기술하여 매칭시에 보다 정확한 매칭이 가능하다.

3. 제안된 시맨틱 웹 서비스 탐색 방법

제안된 방법은 그림 5와 같이 도메인 정보 획득, 서비스 및 사용자 요구 사항의 기술, 그리고 서비스 출판 및 탐색의 3단계로 구성된다. 먼저 도메인 정보 획득 단계에서는 도메인 관계 트리에서 출판 또는 검색하고자 하는 서비스의 도메인을 선택하고, 이를 통해 이 후의 단계에서 사용될 도메인 온톨로지 및 서비스 레이저스트리를 결정한다. 서비스 및 사용자 요구 사항 단계에서는 선택된 도메인 온톨로지의 개념들과 서비스의 입출력 매핑시킴으로써 출판될 서비스 및 사용자 요구 사항을 기술한다. 서비스 출판 단계에서는 온톨로지를 사용하여 기술된 서비스

![Diagram](a) Agency 온톨로지. (b) 개념 기술의 예.

그림 4. 통합 개념 모델의 예

![Diagram](1) Restriction 태입의 예: (Agency, Restriction, (hasName, hasAddress, hasEmail, hasPhone))

![Diagram](2) Partial 태입의 예: (Agency, Partial, (hasName, hasEmail))

![Diagram](3) Coverage 태입의 예: (Agency, Coverage, (hasName, hasAddress, hasEmail, hasPhone, ticketInfo, hasLocation, hasRoom))

그림 5. 제안된 시맨틱 웹 서비스 탐색 방법
스 프로파일을 UDDI에 등록하고, 서비스 탐색 단계 에서는 UDDI에 등록된 서비스 프로파일과 사용자 요구사항 간의 개념 매칭을 사용하여 사용자 요구사항을 만족시키는 서비스를 찾아낸다.

3.1 도메인 정보 획득

제안된 방법에서 새로운 서비스를 등록하거나 원하는 서비스를 탐색하기 위해서는 먼저 도메인 관계 트리로부터 도메인 정보를 획득해야 한다. 도메인 관계 트리의 도메인 간의 관계 및 도메인 온톨로지, 레지스트리 정보가 기술된 트리로서 원하는 도메인의 온톨로지 및 레지스트리 정보를 획득 할 수 있다. 이러한 도메인 관계 트리를 기술하기 위해서는 도메인, 도메인 온톨로지 그리고 레지스트리 간 관계를 기술할 수 있는 방법이 필요하며, 이를 위해 본 논문에서는 그림 6과 같은 레지스트리 온톨로지를 제안한다.

제안된 레지스트리 온톨로지는 단순히 레지스트리 간의 관계를 기술할 기존의 레지스트리 온톨로지에서 온톨로지 및 레지스트리 간의 관계를 함께 기술할 수 있도록 확장한다. 또한 각각의 구성요소에 대한 접근을 효율적으로 지원하기 위한 프로퍼티를 제공한다. 각각의 도메인은 subDomain 프로퍼티를 통해 자신에게 속하는 하위 도메인을 기술할 수 있다. 또한 consistOf 프로퍼티를 통해 도메인에서 사용되는 온톨로지를 기술하고 hasRegistry 프로퍼티를 통해 도메인에 속한 서비스들이 저장되는 레지스트리를 기술한다. 레지스트리는 절의 URI와 출판 URI 두 가지 방식으로 접근할 수 있으므로 inquiryURI와 publishURI 프로퍼티를 구분하여 기술한다. 그림 7은 제안된 레지스트리 온톨로지를 사용하여 기술된 도메인 관계 트리의 예로서, 여행

![그림 6. 제안된 레지스트리 온톨로지](image)

![그림 7. 도메인 관계 트리의 예](image)
(Travel) 도메인에 대한 온톨로지 및 레지스트리 정보를 표현한다.

제안된 방법에서 도메인 온톨로지는 서비스 분류 온톨로지와 데이터 온톨로지 두 개의 온톨로지로 구성된다. 서비스 분류 온톨로지는 도메인에서 제공되는 기능에 대한 분류 정보이며 데이터 온톨로지는 도메인에서 사용되는 데이터에 대한 분류 정보이다. 제안된 방법은 도메인 온톨로지를 서비스 분류 온톨로지와 데이터 온톨로지로 구분함으로서, 복잡한 온톨로지 구조를 파악하기 쉽게 구조화하고 개념 간 매칭이 발생할 수 있는 오류를 줄인다. 그림 8과 그림 9는 각각 여행 도메인에 대한 서비스 분류 온톨로지와 데이터 온톨로지의 예이다. 그림 8은 여행 도메인에서 제공할 수 있는 기능과 이들의 분류 정보를 표현하고 그림 9는 여행 도메인에서 사용되는 데이터와 이들의 분류 정보를 표현한다.

3.2 서비스 및 사용자 요구 사항의 기술

제안된 방법은 서비스와 사용자 요구 사항을 기술하기 위해 OWL-S 서비스 프로파일을 사용한다. OWL-S 서비스 프로파일은 서비스가 어떤 기능을 수행하는지를 기술하며 서비스의 분류, 입력 및 출력, 선형조건(precondition), 서비스 수행 후의 상태 변화(effect) 등의 정보를 제공한다. 제안된 방법은 서비스 프로파일에 포함된 serviceCategory와 hasInput/hasOutput을 사용하여 서비스 분류 온톨로지와 데이터 온톨로지를 기반으로 서비스의 분류 및 입력/출력 정보를 기술한다. 한편 제안된 통합 개념 모델의 개념 타입과 프로퍼티는 OWL-S 프로파일에서 제공되는 기본 요소만으로는 기술할 수 없다. OWL-S는 추가적인 정보를 기술하고자 하는 경우, ServiceParameter를 확장하여 사용하는 방법을 제공한다. 그림 10과 같이 ServiceParameter를 활용하여 통합 개념 모델을 기술할 수 있는 통합 개념 온톨로지를 제안하고 이를 기반으로 개념 모델에 대한 정보를 기술한다. 따라서 제안된 방법은 기존 OWL-S의 표현 방식에서 제공되고 있는 ServiceCategory, hasInput/hasOutput를 사용함으로써 효율성을 유지하고 있으며 ServiceParameter를 통해 보다 향상된 표현방식을 제공한다.

제안된 통합 개념 온톨로지는 ServiceParameter의 프로퍼티인 sParameter의 지역이 되며 Parameter-Feature를 통해 개념 모델을 기술한다. Parameter-Feature는 CategoryFeature, InputFeature, OutputFeature와 같은 자식 클래스를 가지며 이를 통해 시
그림 10. 제안된 통합 개념 온톨로지

그림 11. 통합 개념 모델에 기반한 서비스 프로파일의 예
그림 11. 통합 개념 모델에 기반한 서비스 프로파일의 예(계속)
서비스 분류 및 입출력 개념에 대한 개념 모델을 기술한다. 또한 ParameterFeature는 개념 모델을 기술하기 위해 hasConceptModel 클래스를 통해 ConceptModel 클래스를 참조하며, ConceptModel은 hasClass, hasConceptType, 그리고 hasProperty 프로퍼티를 통해 개념 모델을 기술한다. 그림 11은 항공 서비스 도메인에서 GetFlightInfo 서비스를 OWL-S 서비스 프로파일과 재양된 통합 개념 온톨로지를 사용하여 기술한 예이다. 그림 11에서 GetFlightInfo는 CategoryFeature를 통해 서비스 분류가 ticket과 airport 프로퍼티를 포함하는 AirService의 Restriction 타입이라는 것을 기술한다. 또한 InputFeature를 통해 서비스의 입력으로 hasAddress, hasCity 프로퍼티를 포함하는Partial 타입의 Airport라는 것을 기술하며, OutputFeature를 통해 서비스의 출력이 hasAircraft, hasCarrier, hasTicket 프로퍼티를 포함하는 Restriction 타입의 Flight이라는 것을 기술한다.

3.3 서비스 출판

제안된 방법은 서비스 출판 시 서비스 프로파일을 UDDI에 등록하기 위해 기존의 OWL-S/UDDI 매핑 방법[16]을 확장하여 사용한다. 본 논문에서는 OWL-S 1.1과 UDDI와의 매핑만을 가정한다. 기존 매핑 방법은 서비스 프로파일 구성 요소에 대해 대응되는 UDDI 데이터 구조가 있는 경우에는 직접 해당 데이터 구조에 서비스 프로파일 구성 요소를 저장하고 그렇지 않은 경우에는 tModel을 사용하여 저장한다. tModel은 특정 데이터의 분류 정보를 저장하는데 사용될 수 있으며 이 경우, businessService의 categoryBag에 tModel로 구분된 KeyedReference의 키 이름, 키 값) 쌍을 통해 개념 정보를 저장한다.

기존 매핑 방법에서는 serviceCategory_tModel, hasInput_tModel, hasOutput_tModel을 사용하여 서비스 분류 및 입출력의 클래스 정보를 UDDI에 저장한다. 그러나 제안된 방법에서는 통합 개념 모델을 사용하여 서비스를 기술하므로 클래스 정보뿐만 아니라 개념 타입과 프로퍼티 정보도 모두 저장해야 한다. 이를 위해 ServiceParameter에 저장되어 있는 개념 타입 및 프로퍼티 정보는 categoryConceptType_tModel, categoryProperty_tModel, inputConceptType_tModel, inputProperty_tModel, outputConceptType_tModel, outputProperty_tModel을 사용하여 UDDI에 저장된다. 그림 12는 확장된

![그림 12. 제안된 OWL-S/UDDI 매핑 방법](image-url)
OWL-S/UDDI 매핑을 보여준다. 또한 그림 13은 제안된 OWL-S/UDDI 매핑 방법에 의해 그림 11의 서비스 프로파일이 UDDI에 저장된 예를 보여준다.

3.4 서비스 탐색

제안된 탐색 방법은 그림 14와 같이 사용자 요구사항과 서비스 명세 간의 서비스 분류 매칭과 서비스 데이터 매칭의 두 단계로 이루어진다.

서비스 분류 매칭 단계에서는 서비스 분류 정보에 대한 개념 간 매칭을 통해 서비스 분류 유사도를 계산한다. 서비스 데이터 매칭 단계에서는 업력출 개념 간 단순 및 복합 매칭을 수행하여 출력 유사도와 입력 유사도를 계산한다. 두 단계를 통해 계산된 서비스 분류 만족도, 출력 만족도, 그리고 입력 만족도는 식(1)과 같이 사용자 요구사항 만족도를 계산하는데 사용된다. 식(1)에서 REQcategory와 ADVCcategory는 각각 사용자가 요구하는 서비스의 분류 개념과 등록된 서비스의 분류 개념을 나타낸다. 또한 REQouts와 ADVouts는 출력 개념의 집합을 나타낸다, REQins와 ADVinss는 입력 개념의 집합을 나타낸다. 제안된 방법은 사용자 요구사항 만족도가 임계값보다 큰 서비스를 사용자의 요구

그림 13. 그림 11의 서비스 프로파일을 UDDI 레지스트리에 출현한 예

그림 14. 제안된 웹 서비스 탐색 방법
사용자를 만족하는 서비스로 간주하며, 임계값을 넘는 서비스들을 사용자 요구 사항 만족도의 크기대로 순위를 매겨 반환한다.

사용자 요구사항 만족도(REQcategory, REQouts, REQins), (ADVcategory, ADVouts, ADVins)) = \( wc \times \text{분류유사도}(\text{REQcategory, ADVcategory}) + w_0 \times \text{결과유사도}(\text{REQouts, ADVouts}) + w_1 \times \text{입력유사도}(\text{REQins, ADVins}) \)

\( wc + w_0 + w_1 = 1 \)

(1)

3.4.1 서비스 분류 매칭

서비스 분류 매칭은 찾고자 하는 서비스 기능에 근접한 후보 서비스 검색을 찾는데 사용된다. 서비스 분류 매칭은 개념 간 매칭을 사용하여 식 (2)와 같이 분류유사도를 계산한다. 분류유사도는 사용자가 요구하는 분류 개념을 후보 서비스와 얼마나 만족하는지 평가하며 분류유사도가 임계값 이상일 경우 후보 검색에 포함시킨다.

\[
\text{분류유사도}(\text{REQcategory, ADVcategory}) = \text{결과유사도}(\text{REQcategory, ADVcategory}) \]

(2)

분류유사도는 사용자가 요구하는 개념을 충만된 개념이 얼마나 만족시키는지 평가하기 위해 개념유사도를 사용한다. 개념유사도는 포함 관계가 존재하는 클래스 간 유사도를 계산한다. 개념유사도는 프로퍼티 정보를 기반으로 계산되며 사용자 요구 사항의 개념 Req와 서비스의 개념 Adv에 대해 표 3과 같이 개념 간 포함 관계에 따라 다르게 계산된다.

두 클래스가 동일한 경우 또는 한 클래스가 다른 클래스의 자식 클래스인 경우에는 식 (3)을 통해 개념유사도를 계산한다. 한 클래스가 다른 클래스를 포함하는 경우에는 식 (4) 또는 식 (5)를 통해 개념유사도를 계산한다. 일반적으로 개념 트리에서 개념 간 거리가 멀어질수록 유사성이 떨어지므로, 식 (4)와 식 (5)에서는 식 (3)으로 계산되는 개념유사도에 두 개념 사이의 거리를 지수를 weight 값을 곱하여 거리가 멀수록 유사도가 낮게 계산되도록 한다. 한편 클래스 Adv가 클래스 Req를 포함하는 경우에는 클래스 Adv가 클래스 Req의 프로퍼티를 충분히 제공하지 못하기 때문에 식 (5)에서는 식 (4)보다 낮은 weight 값을 적용한다.

3.4.2 서비스 데이터 매칭

서비스 데이터 매칭은 매칭 방식에 따라 출력 매칭과 입력 매칭으로 구분한다. 출력 매칭 시에는 사용자가 요구하는 출력 개념을 서비스가 얼마나 만족시키는지를 평가하며, 사용자가 원하는 출력 정보에 대해 서비스의 출력유사도가 임계값 이상인 경우 매칭된 서비스로 간주한다. 사용자가 요구하는 서비스 Req에 대해 입력의 서비스 Adv의 출력유사도는 식 (6)과 같이 계산된다. 출력유사도는 먼저 사용자가 요청한 각각의 출력에 대해 서비스의 출력 집합과 단순매칭유사도, 복합매칭유사도, 일대매칭유사도를 계산한다. 사용자가 요구한 출력에 대해 가장

표 3. 포함 관계에 따른 개념유사도의 정의

<table>
<thead>
<tr>
<th>포함 관계</th>
<th>개념유사도</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ equals ADV</td>
<td>( \frac{</td>
</tr>
<tr>
<td>ADV is a subClassOf REQ</td>
<td>( \frac{</td>
</tr>
<tr>
<td>REQ subsumes ADV</td>
<td>( \frac{</td>
</tr>
<tr>
<td>ADV subsumes REQ</td>
<td></td>
</tr>
</tbody>
</table>
출력유사도 \( Req, Adv \_o \) =
\[
\sum_{j=1}^{m} \text{Max}(\text{단순매칭유사도}(req_{i}, Adv_{i}), \text{복합매칭유사도}(req_{i}, Adv_{i}), \text{일대일매칭유사도}(req_{i}, Adv_{i})) + \sum_{j=1}^{n} \text{다대일매칭유사도}(Req_{i}, Adv_{i})
\]
\( m - r + s \) \tag{6}

\( Req_{i} \) = 사용자가 요구하는 서비스 \( Req \)의 출력 개념의 집합,
\( Adv_{i} \) = 종합된 서비스 \( Adv \)의 출력 개념의 집합,
\( m \) = 요청한 개념의 개수, \( req_{i} \subseteq Req_{j}, adv_{i} \subseteq Adv_{j} \)
\( n \) = 서비스 개념의 개수, \( Req_{i} = Req_{j} \) 정복일일 또는 일대일 매칭에 참여한 \( Req_{i} \) 원소들의 집합,
\( r \) = 다대일 매칭에 참여한 \( Adv_{i} \) 원소들의 집합, \( s \) = 다대일 매칭의 수

유사한 서비스 출력을 찾기 위해 가장 큰 유사도를 선택한다. 다음으로 여러 개의 출력 요청을 하나의 서비스 출력이 제공하는 경우를 처리하기 위해 다대일 매칭을 수행한다. 서비스의 각각의 출력에 대해 이전 단계의 매칭에 참여하지 않은 사용자 요청 출력들과 다대일 매칭을 수행하고 유사도를 합한다. Req에 대한 Adv의 전체 출력유사도는 하나의 매칭에 대한 평균값으로 계산된다.

입력 매칭은 사용자 요구 사항의 입력과 서비스의 입력 간에 매칭 정도를 계산한다. 그러므로 입력 매칭은 정보를 제공하는 쪽이 사용자가 되고 정보를 제공하는 쪽이 서비스가 되므로 출력 매칭과 반대로 매칭을 수행한다. 매칭 시에는 서비스가 요구하는 입력 정보를 사용자가 얼마나 제공할 수 있는지 평가한다. 입력유사도는 식 (7)과 같이 계산된다.

\[ \text{입력유사도}(Req_{i}, Adv_{i}) = \text{출력유사도}(Adv_{i}, Req_{i}) \tag{7} \]

입출력유사도에서는 단순 및 복합 매칭을 사용한다. 단순 매칭은 개념 간 일대일 매칭을 수행하여 가장 높은 개념유사도를 갖는 매칭을 선택하고 그 유사도를 반환한다. 예를 들어, 그래프 15와 같이 사용자가 출력으로 Car를 요구하고 서비스가 Car와 Truck을 출력으로 제공하는 경우, 단순 매칭은 Car와 Car, Car와 Truck에 대해 일대일 매칭을 수행하여 개념 유사도가 더 높은 Car와 Car의 매칭을 선택하고 이 때의 유사도를 반환한다. 단순매칭유사도는 식 (8)와 같이 계산된다. 복합매칭은 또한 관계에 기반해서 사용자 요구 사항의 개념 태일이 Coverage 태일이 있고 출판된 개념이 1사 개념일 경우, 일대일 매칭을 수행한다. 예를 들어, 그림 16과 같이 사용자 요구 사항 개념이 Agency이면서 개념 태일이 Coverage 태일인 경우, 사용자는 AirAgency와 CarAgency 그리고 HotelAgency에 대한 모든 정보를 요구한다고 해석할 수 있다. 서비스가 AirAgency와 CarAgency를 제공하는 서비스만 경우, 단순 매칭을 수행하면 두 가지 개념을 제공할에도 불구하고 두 개념 중 하나만 매칭이 된다. 따라서 매칭 정도를 원바르게 평가하려면 사용자의 Agency 요구에 대하여 AirAgency 및 CarAgency 간에 일대일 매칭을 수행하는 것이 옳다. 복합매칭유사도는 식 (9)와 같이 계산되며 사용자가 요구한 개념에 대하여 매칭된 서비스 개념의 비율을 계산한 후 사용자가 요구한 프로퍼티 정보 중 서비스에서 제공되는 프로퍼티의 비율을 곱한다.

![그림 15. 단순 매칭의 예](image-url)

단순매칭유사도 \( \text{req}, Adv \_o \) =
\[
\text{Max}(\text{개념유사도}(\text{req}, adv_{i}), \ldots, \text{개념유사도}(\text{req}, adv_{i}), \ldots, \text{개념유사도}(\text{req}, adv_{m}))
\]
\( \text{req} = \text{사용자가 요구한 개념, Adv} = \text{서비스 개념의 집합,} \)
\( m = \text{서비스 개념의 개수, adv} \subseteq Adv \) (단, \( 1 \leq i \leq m \))

(8)
그림 16. 복합 매칭의 예

![그림 16. 복합 매칭의 예](image)

\[
\text{복합매칭우수도}(\text{reg}, \text{Adv}) = \frac{n}{|\text{class}(\text{reg})|} \times \frac{|\bigcap_{1}^{n} p(\text{adv}_{i}) \cap p(\text{reg})|}{|p(\text{reg})|}
\]

\(\text{reg} = \text{사용자가 요청한 개념,} \)
\(\text{Adv} = \text{서비스가 제공하는 개념의 집합,} \)
\(n = \text{매칭된 Adv 원소의 수,} \)
\(\text{adv}_{i} \in \text{매칭된 Adv 원소들의 집합,} \)
\(\text{임의의 개념} C \text{에 대해} p(C) = C \text{가 포함하는 프로퍼티의 집합} \)

일대다 매칭은 사용자가 요구 사항의 개념이 출판된 개념과 프로퍼티 관계를 갖는 경우에 수행한다. 예를 들어, 그림 17과 같이 사용자가 요구한 개념이 Car이고 요구하는 프로퍼티가 color와 name인 경우, Color 개념과 Name 개념을 출력하는 서비스가 있다면 사용자가 요구하는 Car의 color 정보와 name 정보를 서비스가 제공해 줄 수 있으므로 사용자가 요구하는 Car 개념과 서비스가 제공하는 Color와 Name 개념이 매칭되는 것이 타당하다. 그러나 이러한 매칭은 서비스가 제공하는 Color와 Name이 반드시 Car에 대한 정보일 때만 의미가 있다. 따라서 제안된 방법은 서비스 분류유사도가 임계값 이상인 경우에만 프로퍼티 관계에 기반한 일대다 매칭을 수행한다.

일대다매칭유사도는 식 (10)과 같이 계산되며 각 각의 매칭된 개념이 사용자가 요구하는 프로퍼티를 얼마나 제공해주는지를 평가하여 모든 합한 후 그 값을 사용자가 요구한 프로퍼티의 개수로 나누어 평균값으로 계산된다.

한편 출판된 개념이 사용자가 요구 사항의 개념과 프로퍼티 관계를 갖는 경우에는 다대일 매칭을 수행한다. 예를 들어, 그림 18과 같이 Car을 통해 color와 name 정보를 제공하는 서비스가 있고 사용자가

![그림 17. 일대다 매칭의 예](image)

![그림 18. 다대일 매칭의 예](image)
Color와 Name을 요구한다면 서비스는 사용자가 요구하는 Color 정보와 Name 정보를 제공해 줄 수 있으며 사용자가 요구하는 color와 name 개념이 서비스가 제공하는 car 개념과 매칭되는 것이 타당하다. 그러나 이러한 매칭 역시 전술한 일대다 매칭의 경우와 마찬가지로 서비스 분류유사도가 임계값 이 상일 경우에만 적용된다.

다대일매칭유사도는 식 (11)과 같이 계산된다. 임의의 서비스가 제공하는 개념이 사용자가 요구하는 개념을 프로퍼티로서 제공한다 할지라도, 그 개념이 다른 프로퍼티 정보를 많이 포함하는 경우, 사용자가 요구하는 개념과의 유사성을 보장하기 어렵다. 따라서 다대일 매칭의 경우, 서비스의 프로퍼티와 매칭된 사용자 요청 개념의 수를 서비스가 제공하는 전체 프로퍼티의 수로 나눈 값으로 계산된다.

\[
R_{eq} = \frac{\text{요청된 개념의 집합}}{\text{서비스의 프로퍼티의 수}}
\]

\[
\text{ade} = \text{서비스가 제공하는 개념}
\]

\[
\text{기계문자표(Req,ade)) = \frac{\text{매칭된 } \text{Req 원소들의 값}}{\text{p(ade)}}}
\]

식 (11)

4. 실험 결과

실험은 두 개의 도메인에 속하는 다수의 서비스에 대하여 다양한 형태의 질의를 수행한 후 탐색 결과의 정확률과 재현율을 계산하였다. 실험 결과, 제안된 방법은 평균 96.6%의 정확률과 93.8%의 재현율을 보여 기존 연구보다 우수하였다.

4.1 실험 데이터

실험은 표 4와 같이 Travel 도메인에 속하는 378 개의 서비스와 Economy 도메인에 속하는 264개의 서비스를 대상으로 수행되었다. 모든 서비스는 Travel 온톨로지와 Economy 온톨로지를 기반으로 본 논문에서 제안된 통합 개념 모델을 사용하여 OWL-S 서비스 프로파일 형식으로 기술되었다.

제안된 방법의 탐색 성능을 평가하기 위해 전술한 도메인과 서비스 집합을 대상으로 표 5와 같이 Travel 도메인과 Economy 도메인에 대해 총 13개의 질의를

표 4. 실험 데이터

<table>
<thead>
<tr>
<th>사용 도메인</th>
<th>온톨로지 출처</th>
<th>OWL-S 파일 개수</th>
</tr>
</thead>
<tbody>
<tr>
<td>Travel</td>
<td><a href="http://icl.yonsei.ac.kr/Ontology/Travel.owl">http://icl.yonsei.ac.kr/Ontology/Travel.owl</a></td>
<td>378</td>
</tr>
<tr>
<td>Economy</td>
<td><a href="http://icl.yonsei.ac.kr/Ontology/Economy.owl">http://icl.yonsei.ac.kr/Ontology/Economy.owl</a></td>
<td>264</td>
</tr>
</tbody>
</table>

표 5. 실험에 사용된 질의 형식

<table>
<thead>
<tr>
<th>구분</th>
<th>질의</th>
<th>질의 형식</th>
<th>구분</th>
<th>질의</th>
<th>질의 형식</th>
</tr>
</thead>
<tbody>
<tr>
<td>Travel</td>
<td>T1</td>
<td>입력 : Sports(Coverage)</td>
<td>E1</td>
<td>입력 : PhoneNumber(Restriction)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>출력 : Destination(Partial)</td>
<td></td>
<td>출력 : Company(Restriction)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>입력 : Location(Partial)</td>
<td>E2</td>
<td>입력 : Stock(Restriction), Time(Partial)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>출력 : Airport(Restriction)</td>
<td></td>
<td>출력 : BankStatement(Restriction)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T3</td>
<td>입력 : City(Partial), Country(Partial)</td>
<td>E3</td>
<td>입력 : Time(Restriction), SymbolicString(Partial), Asset(Coverage)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>출력 : Hotel(Restriction)</td>
<td></td>
<td>출력 : Value(Coverage)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T4</td>
<td>입력 : AirTicket(Restriction)</td>
<td>E4</td>
<td>입력 : Date(Partial), Stock(Coverage)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>출력 : Flight(Restriction), Airport(Partial)</td>
<td></td>
<td>출력 : Portfolio(Partial), BankStatement(Restriction)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T5</td>
<td>입력 : Surfing(Restriction), Hiking(Restriction)</td>
<td>E5</td>
<td>입력 : Bond(Restriction), Stock, Exchange(Restriction), Date(Partial)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>출력 : Destination(Partial), Hotel(Partial)</td>
<td></td>
<td>출력 : Portfolio(Coverage)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T6</td>
<td>입력 : City(Coverage), Country(Restriction)</td>
<td></td>
<td>입력 : Stock, Exchange(Restriction), Date(Partial)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>출력 : Hiking(Restriction), Surfing(Restriction), Hotel(Coverage)</td>
<td></td>
<td>출력 : Stock(Partial), Value(Partial)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T7</td>
<td>입력 : Area(Restriction), Date(Partial), Date(Partial)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>출력 : Flight(Coverage), AirTicket(Partial)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
사용하여 실험하였다. 귀의 형태에 따라 탐색 성능이 어떻게 달라지는지를 확인할 수 있도록 한 개의 입력과 한 개의 출력을 갖는 단순한 형태의 귀의 예에서부터 다수의 입력과 출력을 갖는 복잡한 형태의 귀의까지 다양한 형태의 귀의를 사용하였다. 실험에 사용된 귀의는 본 논문에서 제안한 통합 개념 모델을 사용하여 OWL-S 서비스 프로파일 형식으로 기술되었다. 귀의1을 통합 개념 모델을 사용하여 OWL-S 서비스 프로파일 형식으로 기술하면 그림 19와 같다.

그림 19. 귀의 T1의 서비스 프로파일 기술의 예
4.2 성능 분석

제안된 방법의 성능을 분석하기 위하여 두 개의 도메인에서 수행된 실험의 결과에 대해 정확률과 재현율을 계산하였다. 한편 제안된 서비스 탐색 방법은 식 (1)을 통해 계산되는 사용자의 요구 사항 만족도가 임계값 이상인 서비스를 사용자의 요구 사항을 만족시키는 서비스로 간주하기 때문에 임계값에 따라 정확률과 재현율이 달라질 수 있다. 따라서 적절한 임계값을 선택하기 위해 그림 20과 같이 임계값에 따른 정확률과 재현율의 변화를 계산하였으며, 정확률과 재현율이 그레프가 교차하는 지점의 임계값을 선택했다. 그러나 실제 실험 환경에서 정확률과 재현율이 정확하게 교차하는 지점의 임계값을 찾는 것은 매우 어렵기 때문에 그레프에서 교차점과 가장 가까운 0.9를 적절한 임계값으로 선택하였다. 이때 정확률과 재현율은 각각 96.6%, 93.8%였다. 임계값이 0.9일 때, Travel 도메인과 Economy 도메인에서 각각의 임계선에 대한 실험 결과의 정확률과 재현율은 그림 21과 같다.

실험 결과, 제안된 방법은 Travel 도메인에서 평균 95.1%의 정확률과 92.7%의 재현율을 보였고, Economy 도메인에서 평균 96%의 정확률과 평균 93.9%의 재현율을 보였다. 표 6은 실험 결과에 대한 오류 내용을 분석한 표이다. 표 6에서 false positive는 시스템이 찾은 서비스 중 잘못 찾아진 서비스의 비율을 나타내며, false negative는 찾아야 할 서비스 중에서 찾지 못한 서비스의 비율을 나타낸다. 일반적으로 false positive가 높아지면 정확성이 떨어지고, false negative가 높아지면 재현율이 떨어진다.

표 6의 false positive는 제안된 방법이 임계값 만족 시키지 못하는 서비스를 반환하여 정확성이 떨어질 수 있음을 보여준다. 실험에서 false positive는 서비스 요구 사항 만족도가 임계값을 넘지만 임계값을 만족 시키지 못하는 서비스가 탐색된 경우이다. 예를 들어, 그림 21의 T3에서와 같이 임계값을 출력으로 Hotel 정보를 요구하는 경우, 제안된 방법은 Hotel의 일부 정보만을 제공하는 서비스들도 탐색의 결과로
표 6. 오류 분석 결과

<table>
<thead>
<tr>
<th>구분</th>
<th>오류 내용</th>
<th>개수(비율)</th>
</tr>
</thead>
<tbody>
<tr>
<td>false positive</td>
<td>절의 프로퍼티를 일부 만족하는 서비스가 탐색된 경우</td>
<td>25개(4.4%)</td>
</tr>
<tr>
<td>false negative</td>
<td>절의 개념과 서비스 개념 사이에 포함 관계가 있지만 유사 개념인 서비스가 탐색되지 않은 경우</td>
<td>36개(6.2%)</td>
</tr>
</tbody>
</table>

반환한다. 이 경우, 제안된 방법은 해당 서비스가 절의를 완전히 만족시키는 서비스로 판단하는 것이 아니라 사용자 절의를 일부 만족시키는 서비스로 간주하고 다른 서비스보다 낮은 순위를 부여한다. 그러나 기존 방법은 이렇게 절의의 일부를 만족시키는 서비스도 절의를 완전히 만족시키는 서비스로 평가하여 결과 서비스 결과에서 높은 순위를 부여하기 때문에 서비스 요청자의 관점에서 정확도가 낮다고 볼 수 있다. 이와 같이 제안된 방법은 사용자의 요청을 일부 만족시키는 서비스를 낮은 순위로 반환함으로써 서비스 요청자 관점에서 기존 방법보다 정확하고 유연한 탐색 방법을 제공한다고 할 수 있다.

한편 표 6의 false negative는 절의의 요구 사항을 만족시킬 수 있는 서비스가 존재하지만 제안된 방법이 찾지 못하여 제거될 수 있는 서비스가 존재하지만 제안된 방법이 찾지 못하여 제거될 수 있는 경우를 보여준다. 제안된 방법은 개념 사이의 유사성을 판단하는 근거로서 개념 사이의 포함 관계를 사용한다. 그러나 실제 온톨로지는 두 개념이 포함 관계를 갖지 않지만 사람이 판단할 때 유사한 개념인 경우가 존재한다. 예를 들어, Travel 온톨로지의 Hotel과 HotelInfo는 서로 포함 관계는 없지만 유사한 정보를 나타낼 수 있다. 이러한 경우에 제안된 방법은 절의가 Hotel을 출력으로 요구한 경우 HotelInfo를 출력으로 제공하는 서비스는 절의를 만족시키지 못하는 서비스로 간주하여 결과 서비스 집합에 포함시키지 않는다. 그러나 이러한 문제가 온톨로지가 모호하게 기술되어 있기 때문에 발생하는 문제로서, 온톨로지에서 같은 정보를 표현하는 개념은 하나만 사용하거나 개념 간의 포함 관계를 정확히 기술하면 해결될 것으로 판단된다.

제안된 방법은 통합 개념 모델을 사용하여 개념을 기술하고 서비스를 탐색하기 때문에 기존 방법보다 더 많은 정보의 기술을 요구한다. 또한 서비스의 레지스트리에 등록할 때 추가적인 정보를 저장하고 이 정보를 활용하기 때문에 레지스트리에서 정보를 얻어올 때 추가적인 시간 지연이 발생하는 단점이 있다. 그러나 실제 서비스 탐색 시 레지스트리에서 서비스 집합 정보를 얻어오는데 걸리는 시간의 비율은 극히 작다. 따라서 부가적인 정보의 기술과 저장으로 인해 발생하는 비용보다는 제안된 방법의 정확하고 유연한 서비스 탐색이 갖는 장점이 크다고 할 수 있다. 또한 제안된 방법으로 통합 개념 모델을 적용하여 서비스 및 절의를 자세하게 기술할 수 있다는 것은 사용자에게 서비스 탐색의 정확성을 제공한다는 측면에서 장점이라 볼 수 있다. 사용자는 개념 뿐만 아니라 개념 탐색 및 프로퍼티를 통해 찾고자 하는 서비스를 보다 정확하게 기술할 수 있다. 그러나 사용자 편의성 측면에서는 사용자에게 보다 많은 정보를 입력해야 한다는 점은 단점으로 작용할 수 있다.

4.3 기존 연구와의 비교

기존 연구와의 비교를 위해 주요 연구인 Aggarwal 등의 방법[9]과 제안된 방법을 정량적 및 정성적으로 비교하였다. Aggarwal 등의 방법은 제안된 방법의 레지스트리 방법과 가장 유사한 방법을 사용하기 때문에 실험 결과를 비교하기에 가장 적합한 방법이다. 실험 데이터 집합은 Aggarwal 등의 방법에 서도 동일하게 사용될 수 있는 호환성을 가지며, 결과의 공정성을 위해 탐색 세트에 절의에 대해 적절한 서비스로 판단되는 기준을 제안한 방법과 동일하게 적용하였다. Aggarwal 등의 방법 역시 제안된 방법과 마찬가지로 실험을 통해 서비스 선택 임계값을 결정하였다. Aggarwal 등의 방법에서 임계값에 따른 정확률과 재현율의 변화는 그림 22와 같이, 제안된 방법의 실험에서와 같이 정확률과 재현율 그래프가 교차하는 지점과 가장 가까운 0.9를 임계값으로 선택하여 실험하였다. 실험 결과, Aggarwal 등의 방법은 평균 71.4%의 정확률과 평균 74.1%의 재현율을 보였다.
그림 22. Aggarwal 등의 방법에서 임계값에 따른 정확률과 재현율의 변화

각각의 질의에 대해 0.9를 임계값으로 사용하여 제안된 방법과 Aggarwal 등의 방법을 비교한 결과는 그림 23 및 그림 24와 같다. Travel 도메인에서 제안된 방법은 평균 정확률 95.1%, 평균 재현율 92.7%로 평균 정확률과 재현율이 각각 73.1%와 71.5%로 나타난 Aggarwal 등의 방법보다 성능이 우수했다. Economy 도메인 역시 제안된 방법이 평균 정확률 96%, 평균 재현율 93.9%로서 평균 정확률과 재현율이 각각 68.9%와 76.8%로 나타난 Aggarwal 등의 방법보다 성능이 더 우수했다.

제안된 방법이 기존 연구보다 높은 정확률을 보이는 것은 동합 개념 모델을 기반으로 개념 타입과 프로퍼티 정보를 이용하여 정교한 매칭을 수행하기 때문이다. 특히 기존 연구에서는 서비스가 실제로 제공하는 프로퍼티를 고려하지 않기 때문에 부적합한 서비스를 적합한 서비스로 간주하여 정확률이 떨어진다. 예를 들어, 사용자가 질의 T1처럼 특정 프로퍼티를 가지는 Destination을 출력으로 요구한 경우, 기존 연구는 해당 프로퍼티를 가지지 않는 Destination을 출력으로 가지는 서비스를 적합한 서비스로 간주하여 높은 정확률을 부여한다. 이러한 경우, 기존 연구는 사용자에게 잘못된 정보를 제공할 수 있다.

그림 23. Travel 도메인에서의 실험 결과

그림 24. Economy 도메인에서의 실험 결과
또한 Aggarwal 등은 개념 사이에 포함 관계가 존재하지 않는 경우, 공통 부모 관계를 이용한 매칭을 수행하는데, 이때 관련 없는 개념 사이에 매칭이 이루어지는 오류가 발생할 수 있다. 특히 Travel 도메인에서 질의 T5와 T6, Economy 도메인에서 질의 E6과 같이 질의에 대수의 개념이 사용된 경우, 매칭 오류가 상당히 많이 발생하며 제안된 방법보다 정확도가 떨어지는 결과를 나타냈다. 제안된 방법은 부적절하게 찾아낸 서비스의 수가 기존 연구보다 적 기 때문에 평균적으로 기존 연구보다 높은 정확도를 보였다.

한편 제안된 방법이 기존 연구보다 높은 재현율을 보이는 것은 기존의 매칭 방법에서 외부에 제안된 복합 매칭 방법을 이용하여 서비스를 찾기 때문이다. 사용자가 제공하는 입력과 서비스의 입력 사이에 포함 관계에 기반한 입력에 복합 매칭이 가능한 경우, 제안된 방법은 정확한 매칭 정도를 계산하지만 기존 연구는 매칭시 오류가 발생한다. 예를 들어, Travel 도메인에서 질의 T1의 입력이 Sports 개념이면서 Coverage 타입인 경우, 사용자가 Sports 및 Sports의 하위 개념을 모두 입력으로 제공할 수 있음을 나타낸다. 이때 Sports의 하위 개념인 Surfing과 Hiking을 입력으로 요구하는 서비스가 있는 경우, 제안된 방법은 복합 매칭을 통해 그 서비스를 적합한 서비스로 판단하지만 기존 연구는 복합 매칭을 지원하지 않기 때문에 그러한 서비스를 찾아내지 못한다.

제안된 방법은 개념 간 매칭 수행 시 포함 관계 또는 프로퍼티 관계가 존재하는 경우에만 매칭을 수행한다. 그러나 Aggarwal 등은 개념 사이에 포함 관계가 존재하지 않는 경우, 공통 부모 관계를 이용한 매칭을 수행하는데, 이때 관련 없는 많은 수의 개념과 매칭을 수행하므로 추가적인 시간 소요가 발생한다. 따라서 기존 연구에서 매칭을 수행하는데 소요되는 시간은 제안된 방법보다 더 길 수 있다.

5. 결론 및 향후 연구 방향

웹 서비스의 증가와 더불어 웹 서비스를 보다 정확하게 찾고자 하는 요구가 높아짐에 따라 온톨로지 기반의 시릴텍 웹 서비스 탐색 방법이 제안되었다. 그러나 기존 연구는 온톨로지 개념 해석 시 제한된 수준의 개념 해석 방식을 사용하고 명확한 개념 기술 방법을 제공하지 않기 때문에 매칭 오류가 발생할 수 있다. 본 논문에서는 이를 개선하기 위해 통합 개념 모델에 기반한 시릴텍 웹 서비스 탐색 방법을 제안하였다. 통합 개념 모델은 기존의 클래스 뿐만 아니라 개념 타입을 통해 개념의 범위를 표현하고 프로퍼티를 통해 개념의 특성을 기술하여 보다 정확한 개념 기술이 가능하다. 제안된 방법은 통합 개념 모델에 기반하여 서비스와 사용자 요구 사항을 기술하고 이들간에 정교한 매칭을 통해 서비스를 탐색하는 방법을 제안하였다. 서비스의 사용자 요구 사항 정보의 일대일 관계 뿐만 아니라 포함 관계 기반의 복합 매칭 및 프로퍼티 관계에 기반한 다다리 매칭을 지원하여 보다 정확한 매칭을 수행하였다. 성능 평가를 위해 두개의 도메인에 대하여 다양한 질의를 수행한 결과, 제안된 방법은 평균 95.6%의 정확률과 93.8%의 재현율을 보임으로써 평균 71.4%의 정확률과 평균 74.1%의 재현율을 보인 기존 연구보다 우수하였다.

본 논문에서는 기존 연구 중에서 Aggarwal의 방법과 제안된 방법을 비교하였다. 제안된 방법의 성능 향상을 보다 객관적으로 살펴보기 위해 향후에 기존의 다른 방법과도 비교할 계획이다. 제안된 방법은 통합 개념 모델을 사용하여 기존 방법보다 더 많은 의미 정보를 기술한다. 따라서 처리해야할 데이터의 양이 증가함에 따라 복잡한 서비스 기술을 빠르게 처리하고 정보를 관리하기 위한 연구가 필요하다. 향후에는 제안된 웹 서비스 탐색 방법을 기반으로 효율적이고 빠른 서비스 탐색을 제공하기 위해 P2P 네트워크 기반의 분산 레지스터 기반을 적용하는 연구를 진행할 계획이다. 또한 동일한 기능을 가지는 서비스가 여러 개 존재하는 경우, 지리적으로 가까운 서비스를 찾아주면서 사용자에게 보다 빠른 서비스를 제공해주는 것이 가능하다. 따라서 제안된 방법을 확장하여 지리적으로 가까운 서비스를 적절하게 찾아줄 수 있는 방법에 대해서도 연구할 계획이다.

참고 문헌


신 동 훈
2003년 2월 연세대학교 컴퓨터과학과 졸업(학사)
2005년 2월 연세대학교 컴퓨터과학과 졸업(석사)
2005년 3월 - 현재 연세대학교 컴퓨터과학과 박사과정
관심분야 : Internet Computing, Web Services Composition

이 경호
1995년 2월 연세대학교 전산과학과 졸업(학사)
1997년 2월 연세대학교 컴퓨터과학과 졸업(석사)
2001년 2월 연세대학교 컴퓨터과학과 졸업(박사)
2001년 3월 National Institute of Standard and Technology (NIST)
객원연구원
2002년 - 현재 연세대학교 컴퓨터과학과 부교수
관심분야 : Internet Computing, Service-Oriented Computing, Multimedia Document Engineering