pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2013.2.5.359

Semantic-based Automatic Open API Composition Algorithm for Easier-to-use Mashups

Lee Yong Ju†

ABSTRACT

Mashup is a web application that combines several different sources to create new services using Open APIs(Application Program Interfaces). Although the mashup has become very popular over the last few years, there are several challenging issues when combining a large number of APIs into the mashup, especially when composite APIs are manually integrated by mashup developers. This paper proposes a novel algorithm for automatic Open API composition. The proposed algorithm consists of constructing an operation connecting graph and searching composition candidates. We construct an operation connecting graph which is based on the semantic similarity between the inputs and the outputs of Open APIs. We generate directed acyclic graphs (DAGs) that can produce the output satisfying the desired goal. In order to produce the DAGs efficiently, we rapidly filter out APIs that are not useful for the composition. The algorithm is evaluated using a collection of REST and SOAP APIs extracted from ProgrammableWeb.com.

Keywords : Composition Algorithm, Operation Connecting Graph, Ontology Learning, Mashup, Open API

Easier-to-use 매쉬업을 위한 시맨틱 기반 자동 Open API 조합 알고리즘

이 용 주†

요 약

매쉬업은 공개된 Open API를 이용하여 두 가지 이상의 서로 다른 자원을 섞어서 완전히 새로운 서비스를 만드는 웹 애플리케이션이다. 지난 몇 년간 매쉬업에 대한 관심도가 매우 높아 졌지만 수많은 API들을 매쉬업 속으로 결합할 때 여러 가지 이슈들이 존재한다. 특히, 조합 가능한 API들이 매쉬업 개발자에 의해 수동으로 통합될 때 이는 더욱 심각해진다. 본 논문에서는 Open API 자동 조합을 위한 하나의 새로운 알고리즘을 제안한다. 제안된 알고리즘은 오퍼레이션 연결 그래프 구축 및 조합 후보군 탐색으로 구성되어 있다. 우리는 Open API 입출력 사이의 시맨틱 유사도를 기반으로 오퍼레이션 연결 그래프를 구축하고, 원하는 목표를 만족하는 출력을 산출할 수 있는 사이클 없는 방향성 그래프(DAG)를 생성한다. 또한, DAG들을 효율적으로 생성하기 위해 조합에 도움이 되지 않은 API들은 사전에 신속히 필터링되는 전략을 수립한다. 본 논문에서 제안된 알고리즘은 ProgrammableWeb.com 사이트로부터 REST와 SOAP API 집합을 다운로드 받아 실험 분석을 수행하였다.

키워드 : 조합 알고리즘, 오퍼레이션 연결 그래프, 온톨로지 학습, 매쉬업, Open API

1. 서 론

매쉬업(mashup)은 공개된 Open API(Application Program Interface)를 이용하여 두 가지 이상의 서로 다른 자원을 섞어서 완전히 새로운 가치의 서비스를 만드는 것이다. 이는 기존의 자원을 활용하여 컨텐츠를 만들기 때문에 새로운 서비스를 구축하기 위한 개발 기간이나 비용을 상당히 절약할 수 있다. 현재 매쉬업은 미래 IT 융합 서비스의 효과적인 구현 방법으로써 그 관심도가 점점 높아지고 있으며, 그들의 활용도 엔터프라이즈 비즈니스 분야로부터 과학적 e-사이언스 분야에 이르기 까지 매우 다양하다[1]. 예를 들면, Open API와 매쉬업을 위한 대표적인 포털 사이트인 ProgrammableWeb.com에서는 2012년 12월 26일 현재 8247개의 Open API와 6876개의 매쉬업을 제공하고 있다.

그러나 현재까지 Open API들을 조합하기 위한 보편적인 기술로 매쉬업이 널리 알려져 있으나 아직까지 고려해야할

※ 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(No. 2010-0008303). 또한, 이 논문은 2012학년도 경북대학교 학술연구비에 의하여 연구되었음.

† 종신회원 : 경북대학교 컴퓨터정보학부 교수

논문접수 : 2013년 1월 7일
수 정 일 : 1차 2013년 2월 14일, 2차 2013년 2월 19일
심사완료 : 2013년 2월 19일

* Corresponding Author : Lee Yong Ju(yongju@knu.ac.kr)

이슈들이 많이 있다. 첫째, 대부분의 포털 사이트들은 매쉬업을 위한 수많은 Open API들을 제공하고 있지만, 이들을 탐색하고 결합하는 작업이 대부분 수동으로 이루어지기 때문에 이는 매우 힘들고 많은 시간이 소비되는 작업이다. 따라서 매쉬업 개발자들은 원하는 API들을 신속히 발견할 수 있고 어려운 프로그래밍 기술 없이 이들을 쉽게 조합할 수 있는 기법을 필요로 하고 있다. 둘째, 일반적으로 포털 사이트들은 Open API들을 발견하기 위해 단지 키워드 검색과 카테고리 검색만 지원하고 있다. 이들 검색 방법들은 이미 여러 연구에서 밝혀진 바와 같이 나쁜 재현율과 나쁜 정확률 때문에 문제가 많이 있다. 매쉬업 개발을 좀 더 효율적으로 만들기 위해서는 동적으로 API 기능들을 이해할 수 있고 그들 사이의 호환성을 쉽게 파악할 수 있는 시맨틱(semantic) 기반 접근방법이 요구된다. 셋째, 대부분의 매쉬업 개발자들은 최종 매쉬업을 생성하기 위해 필요로 하는 모든 중간단계를 자동적으로 생성해 주기를 원하고 있다[2]. 즉, 현존하는 매쉬업과 결합 가능한 조합 후보들을 개발자들에게 자동적으로 제공해 줄 수 있는 하나의 기반 기술이 필요하다.

본 연구에서는 이러한 이슈들을 해결하기 위해 시맨틱 기반의 자동 Open API 조합 알고리즘을 제안한다. 간혹 사용자들은 API 저장소(repository)로부터 그들의 요구사항을 만족하는 API를 바로 발견(discovery)할 수도 있으나, 보통은 하나의 API 그 자체만으로는 다양하고 복잡한 사용자 요구사항들을 충분히 만족시켜 줄 수가 없기 때문에 API 조합(composition)을 통해 새롭고 유용한 솔루션을 만들 수 있다. 하지만, 이러한 작업 과정에서 발견이란 조합에 포함된 API의 개수가 단지 한 개뿐인 조합의 특수한 경우로 볼 수 있기 때문에 탐색과 조합은 하나의 단일 문제로 취급할 수 있다.

본 논문에서는 먼저 Open API의 특성들을 신택틱(syntactic)하게 정의하고, 시맨틱하게 묘사할 수 있는 하나의 온톨로지 학습 방법을 소개한다. 그리고 이러한 신택틱/시맨틱 정보들을 활용한 Open API 조합 알고리즘을 기술한다. 제안된 알고리즘은 오퍼레이션 연결 그래프(operation connecting graph) 구축 및 조합 후보군(composition candidates) 탐색 과정으로 구성되어 있으며, 본 알고리즘의 최종 결과물은 사용자 요구사항을 만족시킬 수 있는 사이클이 없는 방향성 그래프(DAG: Directed Acycle Graph)의 생성으로 구현될 수 있다. 여기서 DAG들은 전방향/역방향 API 체이닝(chaining)에 의해 점차적으로 생성되는데, 특히, 이들을 효율적으로 생성하기 위해 최종 결과물에 도움이 되지 않은 API들을 사전에 신속히 필터링하는 전략을 수립한다.

2. 관련 연구

지금까지 API 조합 자동화에 관한 연구들은 대부분 웹 서비스(web service) 조합 문제에 집중되어 왔다. 웹 서비스 조합이란 다양한 플랫폼 상에서 개발된 소프트웨어 컴포넌트들의 매쉬업이라 할 수 있고, 목표 요구사항을 지원하기 위한 관련된 웹 서비스들의 오케스트레이션(orchestration)이라 할 수 있다. 이러한 연구를 위해 비즈니스 프로세스 관리(business process management), 시맨틱 온톨로지 조합 언어(예, OWL-S), 그리고 인공지능(예, planning)과 같은 다양한 기법들이 제안되어 왔다. 그렇지만 이러한 기법들은 본 논문과 같이 SOAP, REST, JavaScript, XML-RPC, 그리고 Atom과 같은 다양한 Open API 프로토콜은 다루고 있지 않다.

자동화된 API 조합을 위해서는 유사 매칭 API들을 발견하기 위한 시맨틱 온톨로지 사용이 필수적이다. 온톨로지 구축 방법은 크게 두 가지 방향으로 추진되고 있다. 첫째는 전문가의 수작업으로 시맨틱 정보를 주석처리(annotation)하여 온톨로지를 구축하는 방법이다(예, OWL-S[3], MSMO[4], SA-REST[5] 등). 하지만 수작업에 의해 구축되는 온톨로지는 시간이 많이 소비되고 날로 증가하는 새로운 API들에 대한 확장성에 문제가 많다. 둘째는 수작업으로 온톨로지를 구축하기 어렵기 때문에 온톨로지 학습 방법에 의해 자동 구축하는 방법이다(예, Naive Bayes[6], 클러스터링[7], 자연언어 처리 기법[8] 등). 그러나 이러한 방법들은 SOAP 기반 웹 서비스를 위한 방법들이었고 다양한 프로토콜이 존재하는 Open API를 위해서는 잘 맞지 않는다.

시맨틱 매칭 기법은 자동 조합 문제에 있어서 핵심 이슈이다. Paolucci 등[9]은 질의 입력을 사용하여 원하는 출력을 산출해 낼 수 있는 OWL-S 기반 웹 서비스 매칭 알고리즘을 제안하였다. 이는 본 논문에서 사용한 API 발견 알고리즘과 비슷하다. 그동안 웹 서비스 조합 문제를 해결하기 위해 그래프 기반 알고리즘들이 많이 제안되었는데, Kona 등[10]은 시맨틱 웹 서비스 조합을 위한 간단하고 효율적인 알고리즘을 제안하였다. 이는 첫 단계에서 전방향으로 진행하면서 질의를 만족하는 DAG를 생성한다. Rodriguez-Mier 등[11]은 자동 웹 서비스 조합을 위해 A* 알고리즘을 활용한 휴리스틱 기반 탐색 알고리즘을 제안하였다. Shiaa 등[12]은 OWL-S 기반 웹 서비스 매칭 알고리즘을 활용한 그래프 기반 서비스 조합 알고리즘을 제안하였다. 이러한 작업들은 본 논문의 접근 방법과 비슷하지만 온톨로지 학습 방법을 고려하지 않았고 탐색 속도 향상을 위한 최적 솔루션을 찾지 못하였다.

3. 온톨로지 학습 방법

시맨틱 매쉬업 기술의 성공을 보장하기 위해서는 품질 좋은 온톨로지의 사용이 필수적이다. 하지만 온톨로지 사용의 중요성에도 불구하고 현재 매쉬업을 위한 온톨로지는 거의 존재하지 않으며 이들의 구축도 쉬운 일이 아니다. 이러한 문제는 오늘날 매쉬업의 확산과 발전에 큰 저해 요인이 되고 있다. 본 논문의 온톨로지 학습 방법은 Open API를 개발할 때 자동으로 생성되는 WSDL/WADL/XRDL 문서만

가지고 항목 간 숨어 있는 시맨틱 정보를 찾아내어 온톨로지를 자동 구축하는 것이다. 이는 이미 우리가 제안하였던 Open API 온톨로지 구축 방법[13]을 기반으로 수행된다.

3.1 매개변수 클러스터링 기법

본 절에서는 매개변수(parameter)들에 대해 의미적으로 같은 개념들을 묶는 매개변수 클러스터링 기법을 소개한다. 즉, WSDL/WADL/XRDL 파일에 존재하는 신택틱 정보를 이용하여 그들 사이에 숨겨져 있는 시맨틱 개념(concept)을 얻기 위해 마이닝(mining) 알고리즘을 적용한다. 주된 아이디어는 용어(term)들의 발생 빈도수를 측정하여 용어들을 개념들의 집합 속으로 클러스터(cluster)하는 것이다. 이러한 개념 클러스터는 Open API 사이의 유사성을 결정하는데 사용된다. 우선, 형식적으로 Open API는 다음과 같이 정의될 수 있다.

정의 1: 하나의 Open API=<Σ, Ω>, 여기서 Σ는 API를 묘사하는 이름과 텍스트 설명이며, Ω는 API에서 제공되는 오퍼레이션들의 집합이다. 하나의 오퍼레이션 O=<X, In, Out>는 이름과 텍스트 설명 X와 입력 In, 출력 Out으로 구성된다. 각 In과 Out에는 다수의 매개변수들이 집합으로 구성될 수 있다.

본 논문에서는 오퍼레이션 매칭 문제에 초점을 맞춘다. 다시 말해, 하나의 API 오퍼레이션이 주어졌을 때 이와 유사한 오퍼레이션 리스트를 반환하는 것이다. 이를 위해 Dong[7]이 제안한 것과 비슷하게 오퍼레이션 텍스트 설명과 매개변수에 숨겨져 있는 시맨틱 개념 정보를 추출한다. 일반적으로 입출력 매개변수들은 다수의 단어가 연결된 복합 단어로 이루어져 있다(이들은 개발자의 명명 습관이나 약어 사용 등으로 매우 다양해 질 수 있다). 예를 들면, 매개변수 ArrivalTimeOfAirplane은 첫 번째 대문자에 의해 {Arrival, Time, Of, Airplane}과 같이 잘 분리될 수 있으나, AppId와 같은 경우에는 POS(part-of-speech), 불용어(stop-word) 필터링, 약어 확장, 그리고 동의어(synonym) 처리 과정을 거친 후 수행된다. 이러한 매개변수 내에 존재하는 용어들을 여러 개의 의미 있는 시맨틱 개념 속으로 클러스터링 시킬 때 우리는 용어들의 발생 빈도수를 고려한다. 이는 만일 용어들이 동시에 자주 나타난다면 그것들은 같은 개념을 나타내는 경향이 있다[7]는 가정 하에 하나의 특별한 연관규칙(association rules)[14]에 따라 만들어 진다.

$T=\{t_1, t_2, ..., t_m\}$을 용어들의 집합이라고 할 때, 연관규칙은 용어 t_i가 일어나면 용어 t_j가 일어난다는 의미로 $t_i \rightarrow t_j$(여기서 $t_i, t_j \in T$)로 표현될 수 있다. 트랜잭션(transaction)은 API 입출력에 나타나는 용어들의 집합으로 볼 수 있으며, 지지도(support)와 신뢰도(confidence)는 $t_i \rightarrow t_j$ 규칙이 얼마나 유용한지를 나타내는 지표로서 사용된다. 지지도는 용어 t_i와 t_j를 동시에 포함하는 트랜잭션의 확률을 표현하며, 신뢰도는 t_i가 주어졌을 때 t_j가 동시에 나타날 트랜잭션의 확률을 나타낸다. 만일 규칙 $t_i \rightarrow t_j$의 지지도와 신뢰도가 최소 지지도와 최소 신뢰도보다 크다면 t_i와 t_j는 밀접하게 연관되어 있다고 말할 수 있다. 연관규칙 탐사 문제는 지금까지 잘 알려진 Apriori[15] 알고리즘에 의해 계산될 수 있다.

본 논문에서는 용어들의 집합 $T=\{t_1, t_2, ..., t_m\}$를 시맨틱 개념들의 집합 $C=\{c_1, c_2, ..., c_m\}$로 전환하기 위해 계층적 결합 클러스터링 알고리즘[16]을 사용한다. 그러나 기존의 클러스터링 알고리즘들은 클러스터 사이의 인접성을 기본으로 저차원의 데이터를 위해 설계되어 있어서, 데이터 차원이 매우 큰 텍스트 문서인 경우 "차원의 저주(curse of dimensionality)" 문제가 대두된다. 이러한 문제를 해결할 수 있는 방법으로써 빈발 패턴 기반 클러스터링 방법이 적용될 수 있는데, 이 아이디어는 발견된 빈발 패턴이 또한 클러스터링을 의미하기도 한다는 것이다. 즉 고차원 용어 벡터 공간을 클러스터링하는 대신에 저차원의 용어 집합만을 클러스터 후보로 간주하는 것이다. 따라서 본 논문에서 제안되는 알고리즘은 Apriori 알고리즘의 연관규칙 탐사 결과에서 먼저 신뢰도를 내림차순으로 정렬한 다음 지지도를 내림차순으로 정렬한 후 각 단계에서 가장 최상위에 있는 규칙을 조사하여 만일 두 용어가 다른 클러스터에 속하면 이들을 결합한다.

결합하는 과정에서 가장 이상적인 클러스터를 형성하기 위하여 다음의 두 가지 클러스터 특성을 고려한다. 즉, 클러스터의 응집력(cohesion) - 한 클러스터 내의 용어들과의 응집력 - 은 높게 하고, 클러스터 간의 연관성(correlation) - 다른 클러스터 용어들 간의 상호관계 - 는 낮게 한다. 최종적으로 전체적인 클러스터 품질을 측정하기 위한 클러스터링 점수는 (응집력)/(연관성) 방식으로 계산되며, 이때 우리의 목표는 가장 높은 점수를 갖도록 클러스터를 형성하는 것이다. 이러한 과정은 유사한 클러스터를 병합한 후 새로 계산된 점수가 이전보다 더 높은 점수를 갖는지 조사하고, 더 이상 높은 점수를 얻을 수 없을 때까지 이 과정을 반복한다.

3.2 매개변수 패턴 분석 기법

매개변수 패턴 분석 기법은 매개변수 내에 포함된 용어들 간의 상관관계를 취득하고, 만일 두 용어들이 서로 유사하며 상관관계가 조건에 일치한다면 그 매개변수를 매치하는 것이다. 이러한 접근방법은 사람들이 다수의 용어들을 가지고 매개변수를 만들 때 일반적으로 비슷한 패턴을 사용한다는 관찰로부터 유도되었다. 이러한 패턴들을 조사하기 위해 본 논문에서는 Open API의 대부분을 차지하는 REST, SOAP, JavaScript, XML-RPC API들에 대해 실험 데이터를 만들어 분석하였다. 하지만 JavaScript는 WSDL 파일을 생성하므로 SOAP API에 포함시킬 수 있고, XML-RPC는 차지하는 비율이 낮아 분석에서 생략하였다.

먼저 REST API는 Open API 및 매쉬업 저장소인 ProgrammableWeb.com으로부터 REST API 매개변수들을

다운로드 받아 사전 작업 처리과정을 거쳐 형태소 분석을 수행할 수 있는 데이터 파일을 준비하였다. 이 사이트는 본 논문의 실험 시점에 8247개의 Open API가 존재하고 있었으며, 이들 중 REST 방식으로 구현된 API는 5319개였다. 본 조사에서는 이들 모든 API들을 다 이용하지 않고 mapping, travel, weather 도메인에 있는 168개의 REST API들에 대해서만 실험을 수행하였다. 다음으로 SOAP API에 대해서는 SOAP 관련 웹 서비스 저장소인 xmethods.net에서 WSDL 파일을 다운로드 받아 실험 분석을 수행하였다. xmethods.net에서는 약 600여개의 WSDL 파일이 존재하고 있으나 본 조사에서는 zip, weather, address에 관한 50개의 파일에 대해서만 실험을 수행하였다.

이들 실험 데이터에 POS 형태소 분석기를 적용시킨 결과 REST API에서는 단지 하나의 토큰으로 구성된 매개변수(예, City)가 전체의 43%로 가장 많았고, 명사+명사(30%), 형용사+명사(9%), 동사+명사(7%), 명사+명사+명사(6%), 명사+전치사+명사(5%), 그리고 기타(0.3%) 순으로 나타났다. SOAP API에서는 단지 하나의 토큰으로 구성된 매개변수는 38%를 차지하였으며, 명사+명사(37%), 명사+명사+명사(14%), 동사+명사(6%), 명사+전치사+명사(3%), 형용사+명사(1%), 그리고 기타(1%) 순으로 나타났다. 본 결과로부터 특이한 사항은 다른 프로토콜을 사용하거나, 다른 도메인을 선택하더라도 매개변수 패턴은 단지 출현 빈도의 순위에 다소간의 변동이 있을 뿐 도출된 5개의 패턴 종류는 동일한 사실을 알 수 있다.

Table 1. Ontology Conversion Rules

No	Pattern	Relationships
1	$Noun_1$+$Noun_2$	Parameter propertyOf $Noun_1$
2	Adjective+Noun	Parameter subClassOf Noun
3	Verb+Noun	Parameter subClassOf Noun
4	$Noun_1$+$Noun_2$+$Noun_3$	Parameter propertyOf $Noun_1$
5	$Noun_1$+Preposition+$Noun_2$	Parameter propertyOf $Noun_2$

위의 관찰로부터 매개변수에 대한 온톨로지 변환 규칙은 Table 1과 같이 5개의 규칙으로 정할 수 있다. 명사$_1$+…+명사$_n$의 형태인 경우(No 1, 4) 중심어는 주로 명사$_1$로 표현되며, 중심어와 전체 단어 사이의 관계는 일종의 소유(property) 개념과 많이 닮았다[17]. 형용사+명사, 동사+명사의 형태인 경우(No 2, 3) 중심어는 뒤에 있는 명사로 표현되고 복합단어는 중심어로부터 대부분 시맨틱을 상속(subClass) 받는다[17]. 명사$_1$+전치사+명사$_2$의 형태에서는(No 5) 중심어가 명사$_2$가 되고 중심어와 복합단어 간의 관계는 명사$_1$+…+명사$_n$과 같이 property 관계가 설정된다. 위와 같은 규칙을 사용하여 온톨로지가 구축되고 나면, 다음 단계는 질의문에 의해 두 개념 간 매칭을 시키는 것이다.

정의 2: 두 개의 온톨로지 개념이 다음 조건을 만족하면 매치된다: (1) 어떤 개념이 다른 개념의 속성일 경우(예, CompanyID propertyOf Company), (2) 어떤 개념이 다른 개념의 자식관계인 경우(예, virtualAccount subClassOf Account).

정의 2로부터 우리는 매개변수 유사성을 기반으로 한 매치를 발견할 수 있다. 예를 들면, 매개변수 CityName과 CodeOfCity를 비교한다고 할 때, 키워드 검색은 매치가 실패한다. 왜냐하면 두 개의 매개변수는 일치하지 않기 때문이다(CityName != CodeOfCity). 그렇지만 City가 propertyOf 관계를 가지고 있다면 매칭 점수를 리턴할 것이다. 왜냐하면 이러한 두 개의 매개변수는 CityName propertyOf City와 CodeOfCity propertyOf City의 관계에 의해 서로 밀접하게 연관되어 있기 때문이다.

3.3 API 유사도 측정 기법

본 절에서는 매쉬업을 위한 Open API의 유사도가 어떻게 측정되는지 기술한다. 제안된 온톨로지 학습 방법의 하나의 장점은 신택틱 정보에 추가되는 시맨틱 온톨로지가 API를 정의하는 구조에 큰 변화를 주지 않는 것이다. 따라서 기존의 시맨틱 웹 서비스 매칭 기법들이 API 매칭을 위하여 바로 적용될 수 있다. 본 연구에서는 매칭되는 API들을 효율적으로 발견하기 위해 신택틱과 시맨틱 정보를 혼합 사용하는 방법을 탐구한다. 먼저, WSDL/WADL/XRDL 파일에 작성된 신택틱 정보를 파싱하여 토큰화(tokenization), POS, 불용어 필터링, 약어 확장, 그리고 동의어 탐색을 수행하고, 다음으로 온톨로지 학습 방법에서 구축된 시맨틱 온톨로지를 활용한다.

본 논문에서는 오퍼레이션 유사성을 측정하기 위한 토대로 개념들을 클러스터링하였다. 여기서 오퍼레이션은 벡터 O=<X, In, Out>로 정의되었으므로, 두 개의 오퍼레이션이 주어졌을 때 전체 유사도는 각각의 개별 벡터 컴포넌트들에 대한 유사도의 합으로 결정된다.

먼저, 텍스트 정보(X)에 대한 유사도를 측정하면, 이는 전통적으로 IR(Information Retrieval) 분야에서 널리 사용되고 있는 TF/IDF 방법[18]을 사용하면 쉽게 계산될 수 있다. 다음으로, 입력(In)과 출력(Out) 매개변수들의 유사도를 측정한다. 클러스터링을 고려한 입력은 형식적으로 벡터 $In = < p_i, C_i >$로 묘사된다(출력도 이와 비슷하게 $Out = < p_o, C_o >$로 표현될 수 있다). 여기서 p_i는 입력 매개변수들의 집합이고, C_i는 p_i와 연관된 클러스터링 개념이다. 따라서 입력 유사도는 다음의 두 단계로부터 계산된다(출력도 비슷한 방법으로 처리된다).

- p_i에 대해 위에서 설명한 전처리 과정을 먼저 수행한다.
- 이로부터 나온 각 용어들에 대해 관련된 개념들을 찾아 교환하고 이에 대한 유사도를 계산한다.

유사도는 다음과 같이 매개변수 쌍들의 평균값으로 계산된다. 하나의 질의와 저장소로부터 매치되는 임의의 후보 오퍼레이션 쌍을 (Q, O)라 하고, Q와 O에는 각각 m과 n개의 매개변수들이 있다고 가정하면,

$$Q = (q_1, q_2, \cdots q_i, \cdots q_m)$$
$$O = (o_1, o_2, \cdots o_j, \cdots o_n)$$

로 표현될 수 있으며 Q의 q_i와 O의 o_j 간의 매치를 고려할 때, 클러스터링 기반 매개변수 유사도는

$$Sim(Q, O) = \sum_{i=1}^{m}\sum_{j=1}^{n} Match(q_i, o_j)/(m+n)$$

여기서, $Match(q_i, o_j) = \max\{TF/IDF\ method(q_i, o_j)\}$ for all $1 \le i \le m, 1 \le j \le n$.

위 식을 이용하여 오퍼레이션 입력과 출력 유사도를 각각 계산할 수 있으며, 전체적으로 질의와 저장소에 있는 임의의 오퍼레이션 간의 유사도는 다음과 같이 계산된다.

$$Similarity = w_1(Sim_x) + w_2(Sim_i) + w_3(Sim_o)$$

여기서, Sim_x는 텍스트 유사도, Sim_i는 입력 유사도, 그리고 Sim_o는 출력 유사도이며, w_1, w_2, w_3는 각각의 가중치이고 $\sum w_i = 1$이다. 유사도 결과 값은 0과 1 사이의 실수 값을 리턴한다.

한편, 위의 유사도 측정 기법에서는 3.2절에서 설명한 매개변수 패턴 분석 기법은 고려하지 않았다. Table 1을 활용함으로써 검색 결과의 정확률을 향상시킬 수 있다. 즉, 클러스터링만을 사용한 API 유사성 탐색에서는 연관성 높은 단어들을 한 클러스터에 묶어서 단지 동일한 개념처럼 취급할 뿐 용어들 사이의 계층관계를 무시함으로써 사용자의 의도와는 관계없는 매칭(false positives)이 발생할 수 있다. 따라서 매개변수 유사도를 계산할 때 계층관계 온톨로지 개념에 위배되는 매개변수 쌍들을 배제하여 사용자가 만족할 수 있는 품질 좋은 것만 선택할 수 있는 정제과정이 필요하다. 기본적인 아이디어는 매치되는 매개변수 쌍들 중에서 3.2절에서 설명한 패턴 상관관계를 조사하여 부모 클래스가 일치하지 않을 경우 이 쌍들을 검색 결과에서 배제하는 것이다.

4. Open API 조합 알고리즘

4.1 API 조합 문제

하나의 질의와 저장소 내 하나의 API 집합이 주어졌을 때, 저장소로부터 질의 요구사항과 매치되는 하나의 오퍼레이션을 발견할 수 없는 경우, 두 개 이상의 오퍼레이션들을 같이 결합시켜 원하는 목표를 만족시킬 수 있는 오퍼레이션 체인을 탐색해야 하는데, 이러한 것을 API 조합 문제라 한다. 즉, 어떤 오퍼레이션으로부터 생성되는 출력물이 다른 오퍼레이션의 입력으로 받아들여 질 수 있는 체인을 검색한다. 예를 들면, 특정 지역 날씨를 알 수 있는 오퍼레이션을 검색한다고 했을 때 그 결과는 Table 2와 같다.

Table 2. API Composition Problem

O	I/O	Parameter
Q	In	AreaID, Province, CityName
	Out	Weather
O_1	In	AreaCode, NameOfCity
	Out	Latitude, Longitude, Geography
O_2	In	Latitude, Longitude
	Out	Weather

검색 엔진이 질의 요구사항을 만족할 수 있는 하나의 오퍼레이션을 발견할 수 없을 때, 이 엔진은 API 집합으로부터 다수의 오퍼레이션들을 조합할 수 있다. Table 2에서 Q는 입력으로써 AreaID, Province, 그리고 CityName을 제공하고 출력으로써 Weather를 요구하고 있다. 오퍼레이션 O_1은 입력으로써 AreaCode와 NameOfCity를 요구하고 있는데, 여기서 비록 AreaID와 AreaCode는 다른 형태를 취하고 있지만 ID와 Code가 같은 클러스터 개념으로 표현될 수 있기 때문에 같은 시맨틱으로 참조될 수 있다. 또한 CityName과 NameOfCity도 같은 객체(즉, City)의 속성으로 표현될 수 있으므로 같은 시맨틱으로 고려된다. 따라서 Q의 입력을 받아 O_1을 수행할 수 있다. 한편, O_1은 출력으로써 Latitude, Longitude, Geography를 리턴하는데, O_2는 이들 Latitude, Longitude를 입력으로 받아 출력으로 Weather을 리턴할 수 있다. 따라서 후위의 O_2는 전위의 O_1에 의해 산출되는 출력물을 입력으로 사용하여 원하는 질의 결과를 산출한다. 이러한 형태의 Open API 조합 문제는 다음과 같이 정의될 수 있다.

정의 3: 만일 오퍼레이션 O_1이 출력으로써 Out_1을 산출하고 오퍼레이션 O_2는 입력으로써 이 Out_1을 사용할 수 있다면, 이 O_1과 O_2는 조합 가능하다고 할 수 있다. 따라서 API 조합 문제는 저장소로부터 하나의 연속적인 오퍼레이션 체인을 탐색하는 과정으로 정의할 수 있다.

구체적으로 설명하면, 조합의 첫 단계로 선택되어 지는 오퍼레이션들은 질의의 입력만을 사용하여 수행할 수 있어야 하고, 조합의 마지막 단계로 선택되어 지는 오퍼레이션들은 그 출력들이 질의에서 요구되는 출력물을 모두 충족시켜야 한다. 그리고 조합을 구성하는 어떤 전위 오퍼레이션의 출력은 그 다음 후위 오퍼레이션의 입력으로 사용될 수 있어야 한다.

본 연구에서 제안하는 알고리즘은 조합의 자세한 처리 과정은 알 필요 없이 주어진 질의로부터 원하는 목표를 이룰 수 있다. 즉, 매쉬업 개발자는 질의의 형태로 원하는 목표를 간단히 기술한 후 시스템에 처리를 요구한다. 만일 원하는 결과가 하나의 오퍼레이션 출력에 직접 매치될 수 있다면 그 조합 문제는 오퍼레이션 발견 문제로 단순화될 수 있고, 그렇지 않으면 원하는 결과물을 산출할 수 있는 연속적인 오퍼레이션 체인을 탐색하여 문제를 해결한다. 이러한 오퍼레이션 체인은 주어진 질의로부터 생성될 수 있는 하나의 DAG 검색 문제로 볼 수 있다.

4.2 오퍼레이션 연결 그래프 구축

예비 후보 조합군을 재빨리 계산하기 위해 본 연구에서는 사전에 입출력 연결 가능한 오퍼레이션들을 표현한 오퍼레이션 연결 그래프(OCG: Operation Connecting Graph)를 배치(batch) 작업으로 구축한다. 노드들 간의 연결은 보내는 쪽 출력과 받는 쪽 입력 간의 시맨틱 유사성에 기반을 두고 있다. 그래프 구축 알고리즘은 Fig. 1과 같다. 처음에 저장소에 있는 모든 오퍼레이션들에 대해 각각의 노드를 만든다. 그러고 나서 노드들 사이를 연결한다. 각 노드 n_i의 출력이 모든 노드 n_j에 대해 연결 가능한지 유사도를 계산함으로써 체크되는데, n_i의 출력과 n_j의 입력이 시맨틱하게 유사하다면(즉, $Sim(n_i.Out, n_j.In)>0$), n_i에서 n_j로 역방향으로 간선을 연결하고 유사도를 할당한다. 비슷한 방법으로 n_j의 출력과 n_i의 입력에 대해서도 체크하여 간선을 연결한다. 이렇게 만들어진 그래프는 API 조합 문제를 해결하는데 사용되며, 초기의 그래프는 새로운 API가 추가되면 동적으로 수정된다.

```
for all Operations
  n_i = addNode(Operation)
endfor
for each n_i ∈ N
  for each n_j ∈ N
    if Sim(n_i.Out, n_j.In)>0 then addEdge(n_i, n_j)
    if Sim(n_j.Out, n_i.In)>0 then addEdge(n_j, n_i)
  endfor
endfor
```

Fig. 1. OCG Construction Algorithm

4.3 API 조합 알고리즘

본 알고리즘의 기본적인 원리는 질의의 입력을 사용하여 원하는 출력을 산출해 낼 수 있는 오퍼레이션 체인을 찾는 것이다(이러한 체인은 DAG를 생성하는 것으로서 묘사될 수 있다). 이를 위해서 마지막 단계의 오퍼레이션 출력들은 반드시 질의의 출력을 포함하고 있어야만 하고, 첫 단계의 오퍼레이션 입력들은 질의의 입력만을 사용해야 한다. API 조합 알고리즘을 구현하기 위해 본 연구에서는 기존의 API 발견 알고리즘[13]을 확장/발전시켰다. 또한, 이 알고리즘은 주어진 노드로부터 목표 노드까지 최단 거리를 발견하기 위해 BFS(Breath-First Search) 알고리즘[19]을 사용한다. 제안된 알고리즘은 4단계(서브 그래프 탐색, 시작 노드 첨가, 조합 후보군 검정, 그리고 우선순위 계산)로 조합 문제를 해결한다.

1) 서브 그래프 탐색

이 단계에서는 질의에 매치되지 못하는 것들로 확실시 되는 오퍼레이션들을 먼저 필터링시키는 전략을 채택한다. 즉, 질의의 출력 매개변수들을 모두 포함하고 있는 오퍼레이션들을 저장소로부터 탐색한다(이들을 Last 노드들이라 부른다). 그리고 최소한 하나의 질의 입력 매개변수들을 가지고 있는 오퍼레이션들을 탐색한다(이들을 First 노드들이라 부른다). 그 다음, 모든 Last 노드들에 대해 그 노드에 연결된 노드들을 방문함으로써 n-ary 트리들을 생성한다. 이러한 트리는 OCG로부터 First 노드들이 도달할 때 까지 연결되어 있는 모든 노드 및 간선들을 연속적으로 포함시킴으로써 구축될 수 있다. 이를 위해 BFS 알고리즘을 사용하였으며 최종 결과 트리들은 조합 가능한 후보군(composable candidates)이 될 수 있다.

2) 시작 노드 첨가

이 단계에서 각각의 트리에 대해 하나의 Start 노드가 첨가된다. Start 노드는 동적으로 생성되는 하나의 특수한 Dummy 노드로써, 어떠한 입력 없이 질의의 입력을 이 노드의 출력물로 제공하는 노드이다(즉, $O_0=<\varnothing, Q.In>$). 따라서 하나의 가능성 있는 조합 후보를 찾는 것은 Start 노드로부터 시작하여 Last 노드까지 연결된 하나의 DAG를 생성하는 것이다. 이러한 가능성 있는 조합 후보가 발견되었을 때, 다음 단계에서 이 조합에 참여하는 모든 노드 및 간선들에 대해 검정 과정을 수행한다.

3) 조합 후보군 검정

만일 조합에 참여하는 모든 노드들이 원하는 결과를 산출하기 위해 중복없이 연속적으로 수행되어 질 수 있다면 이러한 조합 후보는 타당하다고 볼 수 있다. 이러한 검정 단계는 Start 노드로부터 역방향으로 수행된다. 이 시점에서 First 노드들은 그들의 모든 입력들이 Start 노드의 출력에 의해 제공되는 오퍼레이션들이다. Out_1은 조합에서 First 노드들로부터 산출되는 모든 출력의 합집합이라 하고, In_1은 질의의 입력(즉, Q.In)이라 할 때, Second 노드들의 입력들은 이전 단계에 의해 산출되는 모든 입력들과 질의 입력의 합집합이다(즉, $In_2=Out_1 \cup In_1$). 이 In_2는 다음 단계의 입력으로써 이용 가능하며, 이러한 진행 과정($In_{i+1}=Out_i \cup In_i$)은 Last 노드들이 도달할 때 까지 반복된다. 또한 이렇게 역방향으로 진행하면서 각 단계에서 최적 패스 생성에 도움을 주지 못하는 중복되는 노드들은 제거된다.

4) 우선순위 계산

하나의 DAG가 질의에서 묘사되는 입출력 요구사항을 만족한다면 하나의 조합 후보(composition candidate)로 고려된다. 이는 질의의 모든 출력 매개변수들이 산출될 수 있고 질의의 입력 매개변수들이 전체 또는 일부분 사용되어져야 하는 것을 의미한다. 하나의 조합 후보가 발견되고 나면 유사도를 계산하기 위해 조합에 참여하는 모든 간선들의 유사도 데이터를 수집한다. 유사도는 간선에 첨부되어 있는 모든 유사도 데이터의 평균값으로 계산되고, 조합 후보의 우선순위가 이 유사도에 의해 결정되며, 정렬된 조합 후보군 리스트 중 제일 위에 있는 결과가 사용자에게 가장 좋은 조합으로 추천된다. Fig. 2는 전체적인 API 조합 알고리즘을 설명하고 있다.

```
if Matching(Q.Out, O.Out)=Ø then fail
if Matching(O.In, Q.In)=Ø then fail
for each Last node
    Call BFS-Algorithm
    Create n-ary trees
endfor
for each n-ary tree
   Add Start node
   i = 1, Ii=Q.In
   Generate DAGi
   NOi=NextOperations(i)
   while not Last node
       Outi=∪∀Outputs(NOi)
       Ii+1=Outi∪Ini
       NOi+1=NextOperations(i+1)
       Remove redundant nodes
       i = i+1
    endwhile
endfor
Ranking composition candidates
```

Fig. 2. API Composition Algorithm

5. 실험 분석

본 장에서는 제안된 알고리즘의 성능을 평가하기 위한 실험 및 분석 결과를 설명한다. 실험 분석을 위한 자료는 Open API와 매쉬업 포털 사이트인 ProgrammableWeb.com 사이트로부터 수집할 수 있다. 하지만 이 사이트에서는 현재 WSDL/WADL과 같은 기술 문서는 제공하고 있지 않다. 따라서 간편한 데이터 구축을 위해 SOAP API 기술 문서인 WSDL 파일을 제공하고 있는 xmethods.net으로부터 17개의 오퍼레이션을 다운로드 받아, 이를 파싱한 후 전처리 과정을 거쳐 우리의 알고리즘들이 적절히 수행될 수 있는 데이터 파일을 준비하였다.

하지만 REST API 기술 문서인 WADL 파일을 제공하는 사이트는 현재 우리가 아는 한 거의 없으므로 ProgrammableWeb.com에서 REST API를 제공하고 있는 공급자들로부터 실험에 필요한 자료를 직접 수작업으로 추출하였다. 본 실험에서는 REST API 공급자들로부터 63개의 오퍼레이션을 추출하였으며, 가능한 정확한 자료를 얻기 위하여 날씨, 여행, 매핑, 주소 등 다른 도메인들로부터 다양한 오퍼레이션들을 선택하였다.

이러한 실험 데이터를 이용하여 먼저 오퍼레이션 연결 그래프 구축 알고리즘(Fig. 1 참조)을 실행시킨 결과 Fig. 3와 같은 결과를 얻었다. 본 그래프는 80개의 노드들과 123개의

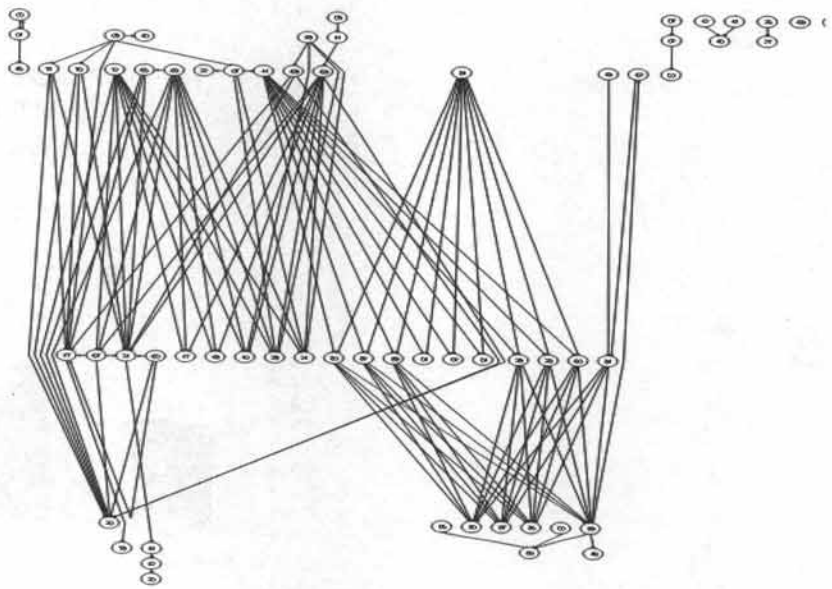

Fig. 3. Operation Connecting Graph

간선으로 구성되어 있으며, 이는 배치 작업으로 수행되어 실시간 API 조합 알고리즘을 효율적으로 지원한다.

본 논문에서는 사전에 구축되어진 오퍼레이션 연결 그래프를 활용하여 API 조합 알고리즘이 수행되어 진다. 다음과 같이 Open API 조합을 위한 하나의 질의 Q가 주어졌을 때,

Q=<{zipcode}, {city, latitude, longitude}>

Fig. 4(A)와 Fig. 4(B)는 본 조합 알고리즘의 결과를 표시한 것이다. 여기서 질의의 입력은 zipcode이고 출력은 city, latitude, longitude이다.

본 알고리즘의 1단계 서브 그래프 탐색 과정으로부터 8개의 Last 노드(짙은 회색 원)와 7개의 First 노드(밝은 회색 원)를 발견하였다(Fig. 4(A)). 그 다음 BFS 알고리즘을 실행시켜서 각 Last 노드들에 대한 n-ary 트리를 생성하는데, 이 과정은 모든 Last 노드들이 도달할 때까지 반복된다(Fig. 4(B)). 여기서 n-ary 트리들은 조합 가능한 후보군이 될 수 있다.

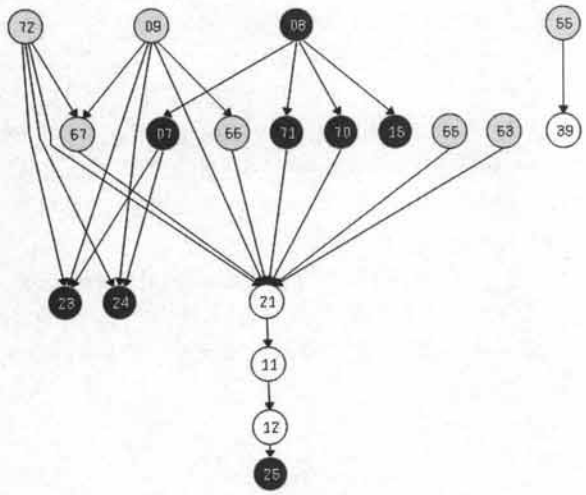

Fig. 4(A). Last and First Nodes

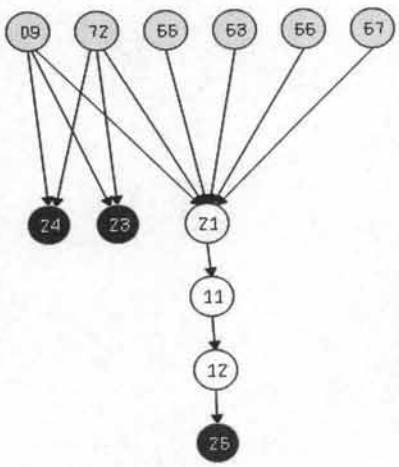

Fig. 4(B). n-ary trees

Fig. 4(B)로부터 가능한 조합 후보가 총 3개 생성되었음을 알 수 있는데, 2단계에서 각각의 트리에 Start 노드 O_0가 첨가된다. 알고리즘은 Start 노드에서 Last 노드까지 DAG를 생성하고, 3단계에서 최적 패스를 생성하기 위한 후보군 검정 과정이 수행된다. 검정 과정이 수행된 후 최종 조합들이 선택되고 지며, 마지막으로 4단계에서 유사도를 기반으로 한 우선순위가 계산된다. Table 3은 이러한 우선순위별로 정렬된 조합 리스트를 보여주고 있다(여기서 결과는 질의 유사도 > 0 것들임).

Table 3. Ranked DAG List

Rank	Score	DAG
1	0.625	O_0 → (9, 72) → 23
2	0.550	O_0 → (9, 72) → 24
3	0.222	O_0 → (9, 72, 65, 66, 67) → 21 → 11 → 12 → 25

최종 조합 결과에 대한 품질을 평가하기 위하여 본 실험에서는 제안된 조합 알고리즘으로부터 원하는 목표가 몇 퍼센트(%) 취득되었는지 수작업으로 체크하였다. Table 3에서 1, 2순위의 조합은 원하는 목표를 만족하는 조합 결과로 판단되었지만 3순위 조합은 사용자 요구사항을 충분히 만족시키지 못하는 잘못된 조합으로 판정되었다. 이러한 결과(즉, 약 67%의 성공률)는 본 알고리즘이 사용자가 요구하는 대부분의 결과를 충분히 산출할 수 있음을 보여주고 있다.

다음으로 본 논문에서 제안하는 온톨로지 학습 방법의 우수성을 분석하기 위해 기존의 IR 분야에서 가장 보편적으로 활용되고 있는 정확률(precision)과 재현율(recall), 그리고 F-점수(F-score)를 사용하여 평가하였다. Fig. 5는 기존의 전통적인 키워드 검색 방법과 클러스터링 기법을 적용한 Woogle 방법[7]을 우리의 온톨로지 학습 방법(Semantic)과 비교하여 제안된 방법의 우수성을 보여주고 있다.

키워드 검색 방법은 예측된 바와 같이 정확률, 재현율, F-점수 모두 가장 낮다. Woogle 방법은 키워드 검색 방법보다 재현율은 개선되었으나 정확률은 거의 비슷하다. 이는 검색된 결과에서 적합한 오퍼레이션들이 증가한 만큼 비례

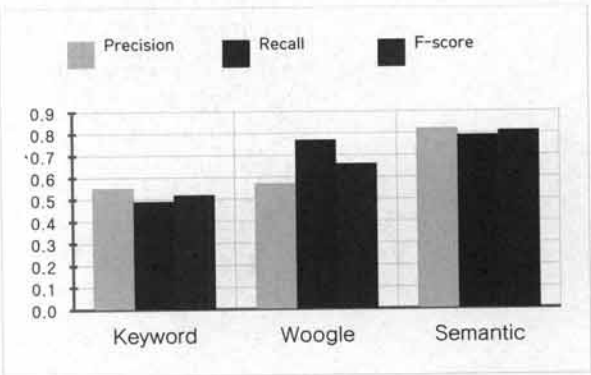

Fig. 5. Precision, Recall, F-score Performance

적으로 부적합한 서비스들도 증가하기 때문이다. 우리의 온톨로지 학습 방법은 검색 결과 중 부적합한 오퍼레이션들을 정제함으로써 정확률과 재현율을 함께 증가시킬 수 있다. 이에 따라 F-점수도 가장 좋은 결과를 얻게 된다.

6. 결 론

본 논문에서는 사용하기 쉬운(easier-to-use) 매쉬업을 구현하기 위하여 자동 Open API 조합 알고리즘을 제안하였다. 본 알고리즘은 그래프 기반 접근방식에 기반을 두고 있으며, 조합 가능한 후보군들이 API 전방향/후방향 오퍼레이션 체인에 의해 점차적으로 생성된다. 또한, 본 알고리즘에서는 질의를 만족시키지 못하는 API들을 재빨리 필터링 시키는 전략을 수립함으로써 최적 패스를 신속히 얻을 수 있다. 이를 위해 제안 알고리즘은 먼저 Last와 First 노드들을 탐색한 후 OCG로부터 n-ary 트리들을 생성하는데, 이러한 서브 그래프 탐색 기법은 자동 조합 알고리즘의 성능 향상에 결정적 역할을 끼쳤다. 한편, 본 논문에서는 API를 신택틱하게 묘사할 수 있는 API를 새로 정의하였고, API를 시맨틱하게 표현할 수 있는 온톨로지 학습 방법을 소개하였다. 이러한 시택틱/시맨틱 정보들은 Open API 조합을 자동화할 수 있도록 지원하고 있다.

향후 연구과제로는 제안된 조합 알고리즘의 성능을 타 시스템과 비교 분석하는 연구가 필요하며, 좀 더 다양하고 많은 질의 환경에서 조합 결과의 품질과 시스템 반응 속도를 평가할 필요가 있다. 또한, 인공 지능 분야의 휴리스틱(heuristic) 플래닝 기법과 동적 최적화 그래프 기법에 관한 연구 결과물들을 본 연구에 접목시킴으로써 우리의 시스템 기능을 보다 최적화시킬 수 있을 것이다.

참 고 문 헌

[1] V. Hoyer and M. Fischer, "Market overview of enterprise mashup tools," in Proceedings of the *6th International Conference on Services Oriented Computing*, 2008, pp.708-721.

[2] H. Elmeleegy, A. Ivan, R. Akkiraju, and R. Goodwin, "MashupAdvisor: A recommendation tool for mashup development," in Proceedings of *the IEEE International Conference on Web Services*, 2008, pp.337-344.

[3] OWL Services Coalition, "OWL-S: Semantic markup for web services," *OWL-S White Paper*, 2004.

[4] T. Vitvar, M. Zaremba, M. Moran, M. Zaremba, and D. Fensel, "SESA: Emerging technology for service-centric environment," *IEEE Software*, Vol.24, No.6, pp.56-67, 2007.

[5] 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, pp.91-94, 2007.

[6] 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.

[7] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in Proceedings of the *30th International Conference on Very Large Data Bases*, 2004, pp.372-383.

[8] 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, pp.340-465, 2005.

[9] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic matching of web services capabilities," in Proceedings of the *First International Semantic Web Conference on the Semantic Web*, 2002, pp.333-347.

[10] K. Kona, A. Bansal, M. Blake, and G. Gupta, "Generalized semantics-based service composition," in Proceedings of the *IEEE International Conference on Web Services*, 2008, pp.219-227.

[11] P. Rodriguez-Mier, M. Mucientes, and M. Lama, "Automatic web service composition with a heuristic-based search algorithm," in Proceedings of the *International Semantic Web Conference*, 2011, pp.81-88.

[12] M. Shiaa, J. Fladmark, and B. Thiell, "An incremental graph-based approach to automatic service composition," in Proceedings of the *International Semantic Web Conference*, 2008, pp.397-404.

[13] Y. J. Lee and J. H. Kim, "Semantically enabled data mashups using ontology learning method for Web API," in Proceedings of the *2012 Computing, Communications and Applications Conference*, 2012, pp.304-309.

[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] R. Agrawal and R. Srikant, "Fast algorithm for mining associations rules," in Proceedings of the *20th International Conference on Very Large Data Bases*, 1994, pp.487-499.

[16] L. Kaufman and P. J. Rousseeuw, *Finding Group in Data: An Introduction to Cluster Analysis*, New York, John Wiley & Sons, 1990.

[17] S. Mokarizadeh, P. Küngas, and M. Matskin, "Ontology learning for cost-effective large-scale semantic annotation of web service interfaces," in Proceedings of the *17th International Conference on Knowledge Engineering and Management by the Masses*, 2010, pp.401-410.

[18] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," *Information Processing and Management*, Vol.24, No.4, pp.513-523, 1988.

[19] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms* (Second Edition), MIT Press, 2001.

이 용 주

e-mail : yongju@knu.ac.kr

1985년 한국과학기술원 산업공학과 정보검색전공(석사)

1997년 한국과학기술원 정보및통신공학과 컴퓨터공학전공(박사)

1998년~현 재 경북대학교 과학기술대학 컴퓨터정보학부 교수

관심분야 : Semantic Web Services, Information Retrieval, Web Databases