복합 웹 서비스를 위한 자연어 인터페이스
(Natural Language Interface for Composite Web Services)

임 종 현 † 이 경 호 ††
(JongHyun Lim) (Kyong-Ho Lee)

요 약 복합 웹 서비스를 이용하고자 하는 사용자의 요구가 늘어나고 있지만 유비쿼터스 환경에서 일반 사용자들이 쉽게 복합 웹 서비스를 사용할 수 있는 환경은 마련되지 않고 있다. 본 논문에서는 웹 서비스에 대한 기술적인 지식이 없는 일반인들도 쉽게 자신의 요구사항을 표현 가능한 자연어 기반의 사용자 친화적인 인터페이스를 제안한다. 특히 사용자의 표현을 효율적으로 분석하기 위해 사용자의 문장으로부터 정교한 수준의 복합적인 워크플로우를 추출하고, 문장 분석을 통해 적합한 서비스를 찾는 방법을 제안한다. 특히 많은 제어 구문과 구절로 이루어진 복잡한 문장으로부터 정교한 수준의 추상 워크플로우를 추출한다.

키워드 : 복합 웹 서비스, 시맨틱 웹 서비스, 자연어 인터페이스

Abstract With the wide spread of Web services in various fields, there is a growing interest in building a composite Web service, however, it is very difficult for ordinary users to specify how to compose services. Therefore, a convenient interface for generating and invoking composite Web services are required. This paper proposes a natural language interface to invoke services. The proposed interface provides a way to describe users' requests for composite Web Services in a natural language. A user with no technical knowledge about Web services can describe requests for composite Web services through the proposed interface. The proposed method extracts a complex workflow and finds appropriate Web services from the requests. Experimental results show that the proposed method extracts a sophisticated workflow from complex sentences with many phrases and control constructs.

Key words : Composite Web services, Semantic Web services, Natural language interfaces

1. 서 론

웹(World Wide Web) 기술의 비약적인 발전으로 인해 웹을 통해서 정교한 수준의 다양한 서비스를 제공할 수 있는 환경이 조성되고 있다. SOAP, WSDL 등 표준에 기반한 웹 서비스[1] 기술은 분산 환경에서 이종 플랫폼 간의 상호운용성을 지원한다. 특히 유비쿼터스 기술의 발전과 모바일 디바이스의 보급으로 인해 사용자가 언제 어디서든 원하는 서비스를 이용할 수 있는 환경이 조성되고 있다. 이에 유비쿼터스 환경에서 모바일 디바이스를 이용하는 사용자가 서비스를 보다 편리하게 호출할 수 있는 방법이 요구된다.

사용자의 요구사항이 그림 1의 요구사항과 같이 복잡할 경우 단일 서비스로 사용자의 요구를 만족시키기 힘들다. 이러할 경우 복합 웹 서비스(composite Web services)를 이용하면 복잡한 요청도 처리 가능하다. 그러나 일반 사용자가 복합 서비스를 구성하는 데에는 많은 어려움이 따르게 된다. 그렇기 때문에 사용자 친화적인 인터페이스를 통한 복합 웹 서비스를 구성하는 연구가 필요하다. 특히 자연어 인터페이스는 사용자가 쉽게 접근할 수 있는 인터페이스의 하나로 이를 이용한 웹 서비스 호출에 대한 연구가 필요하다. 본 논문에서는 그림 1에서와 같이 사용자의 복잡한 질의에 대해 추상 워크플로우(abstract workflow)를 생성하는 시스템을 제안한다.

웹 서비스를 위한 자연어 인터페이스에 대한 기존 연

† 비 회 원 : 연세대학교 컴퓨터과학과
jhlim@icl.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
khlee@cs.yonsei.ac.kr
논문접수 : 2008년 7월 22일
심사완료 : 2009년 11월 1일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제2호(2010.2)

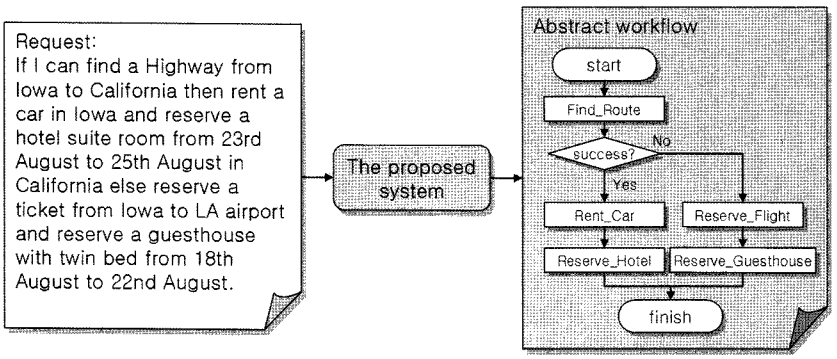


그림 1 자연어 질의에 대한 추상 워크플로우

구는 다음과 같다. Bosca 등[2-4]은 사용자의 질의 문장에서 직접적으로 워크플로우의 형태를 이끌어 내고 사용자의 의도에 맞는 서비스를 찾는 방법을 제안한다. Xie 등[5]은 사용자의 질의를 분석하여 SOBL(Semantic Object Behavior Language) 형식의 언어로 변환한 다음 Semantic UDDI를 참고하여 사용자의 질의에 알맞은 서비스를 찾는 방법을 제안한다. Englmeier 등[6]은 프로세스 디자이너가 비즈니스 프로세스를 자연어 기반의 스토리 북 형태로 입력하면 이를 적절한 비즈니스 프로세스와 매핑하는 WS-Talk 시스템을 제안한다.

Jang 등[7]은 사용자의 질의가 입력되면 해당하는 질의에 대한 최선의 답을 줄 수 있는 웹 서비스를 찾고, 해당 웹 서비스를 실행하여 얻은 결과를 자연어 형태로 변환하여 답변을 하는 방법을 제안한다. Al-Muhamed와 Embley[8-10]은 서비스의 도메인 온톨로지(domain ontology)를 이용하여 사용자의 자연어 요청을 술어계산의 형식으로 변환하여 보다 논리적이면서 형식적으로 사용자의 의도를 표현하는 방법을 제안한다. 기존 연구의 대부분은 간단한 수준의 제어 구문만을 지원하고 있고, 하나의 구절에 대해 단일 서비스 만을 매핑하여 단일 서비스로 제공 불가능한 구절의 경우를 고려하지 않는다.

본 논문에서는 사용자의 자연어 입력으로부터 정교한 수준의 워크플로우를 추출하는 방법을 제안한다. 제안한 방법은 복잡한 워크플로우를 표현한 사용자의 자연어 질의에 대해 기본적인 워크플로우 패턴을 중첩적으로 적용하여 다양한 워크플로우의 형태를 추출한다. 그리고 질의에서 요구하는 서비스의 기능과 대상을 분석하여 적합한 서비스를 찾고, 추출한 워크플로우를 의미적, 구조적으로 분석하여 가장 사용자의 의도에 적합한 워크플로우의 형태를 찾는다. 질의를 수행하기 위한 서비스들과 워크플로우의 형태를 종합하여 사용자의 자연어 질의에 최적화된 추상 워크플로우를 생성한다.

제안한 방법은 다수의 제어 구문과 구절로 이루어진 복잡한 문장으로부터 정교한 수준의 추상 워크플로우를 추출할 수 있다. 그리고 단위 구절에 대응하는 단일 서비스가 없을 경우, 복합 서비스를 구성하여 사용자가 원하는 서비스를 제공한다. 제안된 방법의 성능을 평가하기 위해 다양한 종류의 자연어 문장을 대상으로 실험한 결과, 제안된 방법은 95.2%의 실험 데이터에 대하여 추상 워크플로우를 정확히 추출하였다.

본 논문의 구성은 다음과 같다. 먼저 2절에서는 본 논문에서 제안하는 방법을 적용하기 위해 필요한 온톨로지에 대해 기술한다. 3절에서는 자연어 질의로부터 추상 워크플로우를 생성하는 방법에 대해 자세히 기술한다. 4절에서는 실험을 통해 제안된 방법의 성능을 분석하고, 기존 연구와의 비교를 통해 제안된 방법의 장단점을 기술한다. 마지막으로 5절에서는 결론과 향후 연구 방향에 대해 기술한다.

2. OWL-S 서비스 온톨로지 등록

사용자의 자연어 질의를 분석하여 그에 맞는 서비스를 찾기 위해선 자연어와 서비스를 연결하기 위한 정보가 필요하다. 즉 사용자가 사용하는 단어가 어떠한 서비스를 의미하는지 아니면 어떠한 입출력 값을 의미하는지 알아내야 한다. 이를 위해 서비스의 특징을 의미적으로 기술하고, 어떠한 자연어 단어 또는 문장들이 그 특징과 연관되어있는지 알아야 한다. 또한 서비스에서 요구하는 입출력 값들의 자연어 표현 방식을 미리 구축해야 질의가 들어왔을 때 질의에서 어떠한 입출력 값들이 사용되었는지 분석할 수 있다. 본 절에서는 서비스의 특징을 기술하기 위한 Action 온톨로지와 Object 온톨로지를 제안한다. 그리고 서비스에서 사용하는 입출력 값들을 기술하기 위한 IO 온톨로지에 대해 기술한다.

하나의 서비스에는 서비스에서 사용하는 입출력 값들

이 존재하게 된다. OWL-S[11]에서는 서비스의 입출력 값을 기술할 때 온톨로지상의 컨셉을 사용해 의미적으로 기술한다. 본 논문에서는 이러한 입출력 컨셉들에 대한 온톨로지를 입출력 온톨로지라 한다. 입출력 온톨로지는 알려진 서비스들이 사용하는 모든 입출력 값들의 컨셉들을 포함하고 있으며 이들 입출력 컨셉들 간의 관계를 표현한다. 사실 모든 서비스들에 대해 하나의 통합된 입출력 온톨로지를 구성하는 것은 많은 어려움이 따른다. 그러나 이러한 통합 온톨로지 구성에 대한 내용은 본 논문의 범위를 벗어나므로 본 논문에서는 모든 서비스들의 입출력 컨셉들로 구성된 통합된 입출력 온톨로지를 구성할 수 있는 것으로 가정한다. Action 온톨로지는 서비스의 기능이나 행위에 대한 기술한다. 그리고 Object 온톨로지는 서비스의 대상을 기술한다.

그림 2는 '항공편 검색 서비스'를 기술하는 OWL-S를 등록하는 예를 보여준다. 서비스 제공자는 서비스를 Action 온톨로지의 'Find' 컨셉과, Object 온톨로지의 'Flight' 컨셉과 매핑한다. 그리고 입출력 값들은 입출력 온톨로지의 컨셉들과 매핑된다.

3. 자연어 질의로부터 추상 워크플로우 추출

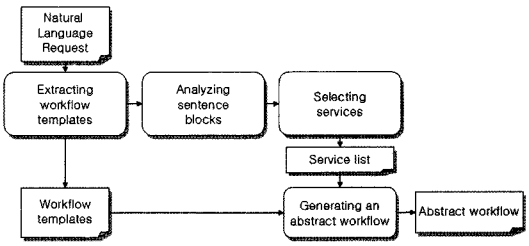


그림 3 자연어 질의로부터 추상 워크플로우 생성 과정

자연어 질의로부터 추상 워크플로우를 추출하기 위해 제안된 방법은 그림 3에서 나타난 바와 같이 네 가지 단계로 이루어진다. 먼저 자연어 질의가 들어오게 되면 제어 구문과 동사를 이용하여 질의를 여러 개의 문장 블록으로 나눈다. 제어 구문의 등장 패턴을 이용하여 추출 가능한 모든 워크플로우 템플릿을 추출한다. 두 번째로 각각의 문장 블록에 대해 서비스의 유형을 분석하여 이에 적합한 서비스 목록을 추출한다. 세 번째로 추출한 서비스 목록과 문장 블록간의 유사도를 계산하여 가장 적합한 서비스를 선택한다. 마지막으로 워크플로우 템플릿과 선택한 서비스에 대한 정보를 결합하여 질의를 만

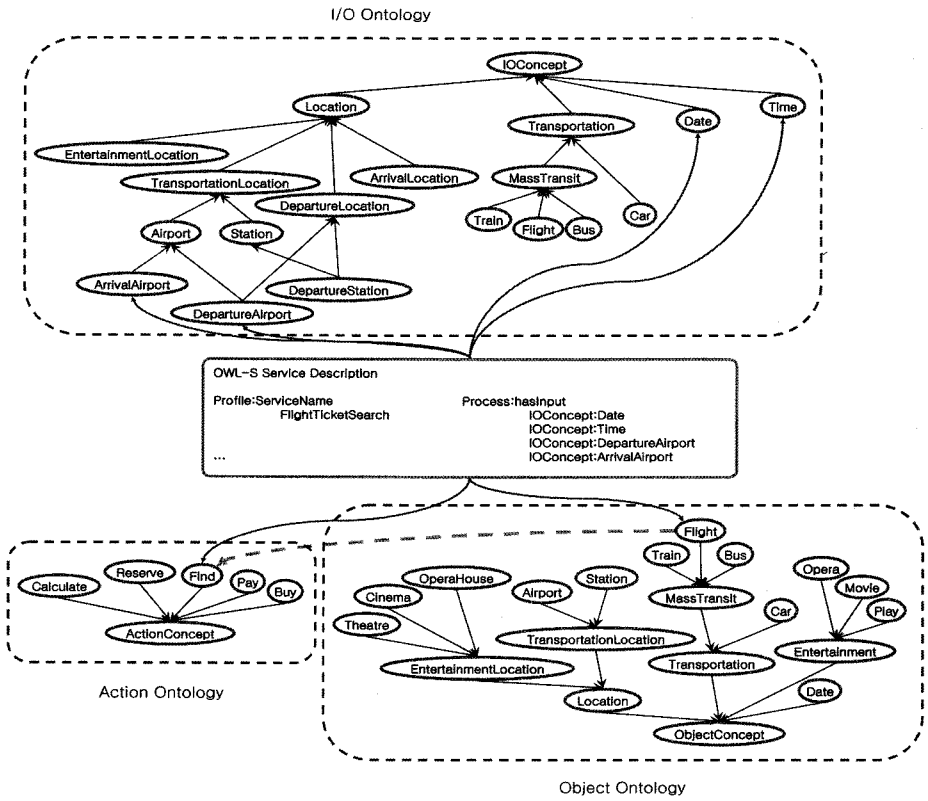


그림 2 서비스와 입출력 및 Action, Object 온톨로지 매핑

족하는 추상 워크플로우를 생성한다.

3.1 워크플로우 템플릿의 추출

사용자가 입력한 자연어 질의는 다수 서비스 호출을 포함할 수 있다. 제안된 방법은 주어진 질의를 동사와 제어 구문을 이용하여 문장 블록들로 나눈다. 그리고 문장 블록들과 제어 구문을 이용하여 워크플로우 템플릿을 추출한다.

그림 4는 본 절에서 워크플로우 템플릿 추출 과정을 나타낸다. 우선 사용자의 요청을 자연어 처리기를 이용해 분석한다. 본 논문에서 자연어 처리기로 기존의 RASP system[12]을 사용하여 문장을 구성하는 단어의 품사 및 단어들간의 관계를 분석한다. 예를 들어, 추출된 접속사에 기반하여 문장의 제어 구조를 파악한다. 제안된 방법은 동사를 이용하여 문장 블록을 나누기 때문에 질의로부터 동사들을 추출한다. 특히 조동사와 부정사를 동반하는 동사와 같이 다른 동사를 수반하거나 보조적인 의미로 사용되는 동사는 제외한다. 추출한 동사와 제어 구문을 이용하여 입력된 질의를 다수의 블록으로 분할한 후, 이를 기반으로 기본적인 워크플로우 패턴을 반복 적용하여 워크플로우 템플릿을 추출한다.

예를 들어, 그림 5와 같이 주어진 질의에서 제어 구문인 'If'와 'then' 등을 추출한 다음 조동사를 제외하고, 'find'와 'reserve', 'rent'동사를 추출한다. 질의에서 제어 구문과 문장 블록의 등장 패턴을 가지고 미리 워크플로우 패턴 리스트에 등록된 패턴들과 비교하여 같은 패턴이 등록되어 있으면 해당 패턴에 해당하는 워크플로우 템플릿을 추출 결과로 선택한다.

워크플로우 템플릿을 이용하여 사용자 질의로부터 추출한 워크플로우가 경우에 따라 다양한 형태가 될 수있다. 그림 6에서 사용한 질의는 다양한 제어 구문을 사

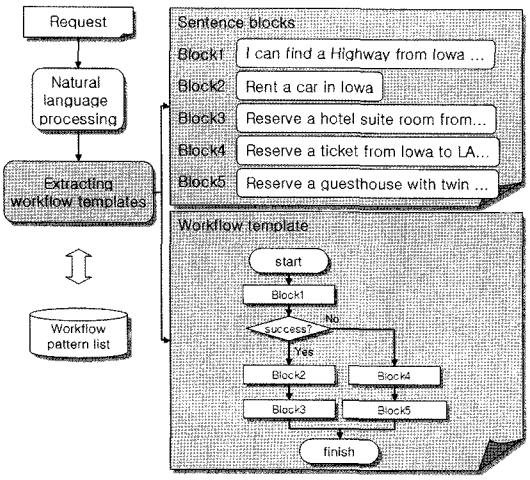


그림 4 워크플로우 템플릿 추출

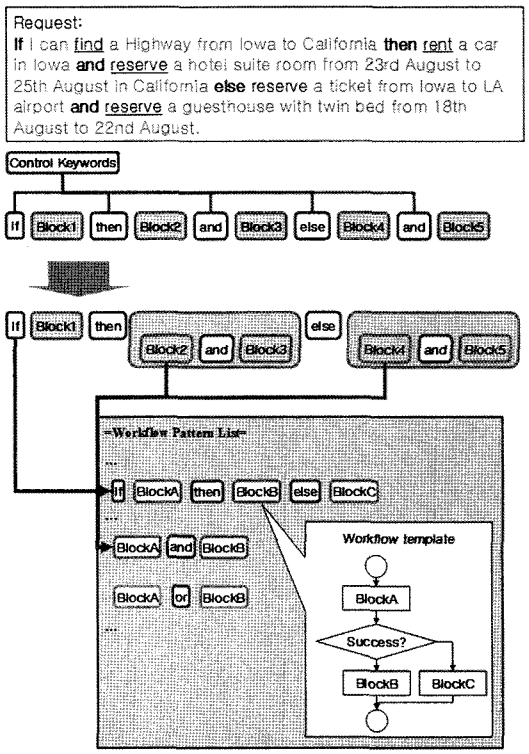


그림 5 워크플로우 추출의 예

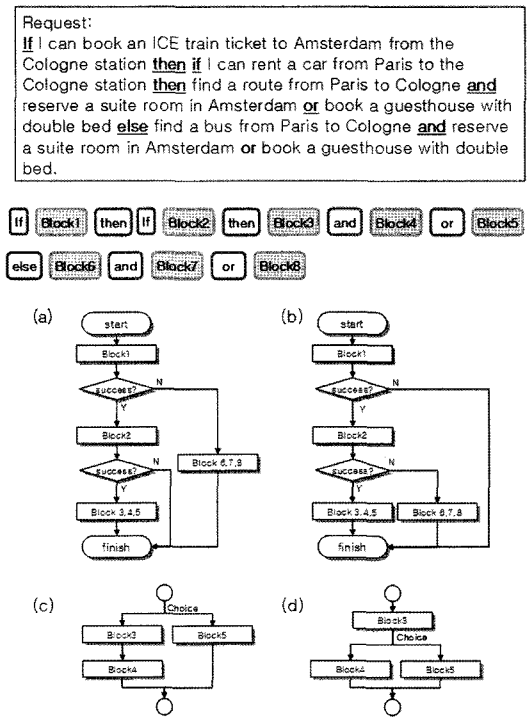


그림 6 워크플로우 다중 분석

용하고 있어서 제어 구문간의 우선순위나 조합 방법에 따라 여러 형태의 워크플로우 템플릿이 나타난다. 'if-then'과 'if-then-else'가 중첩적으로 사용될 경우, 'else' 제어구문이 어떠한 'if-then' 제어구문에 종속적인지 판단하는 방법에 따라 워크플로우 (a)와 (b)의 경우처럼 서로 다른 워크플로우가 생성된다. 그리고 'and'와 'or'가 연속해 나타날 경우, 워크플로우 c)와 d)처럼 서로 다른 워크플로우가 생성이 될 수 있다. 이렇게 하나의 질의에 대해 서로 다른 워크플로우가 생성될 경우, 사용자의 질의를 의미적으로 분석하여 사용자의 의도에 가장 적합한 워크플로우를 도출하는 과정이 필요하다. 이 과정은 3.4절에서 자세하게 기술한다.

3.2 문장 블록 분석

사용자의 자연어 질의는 그 자체로는 문자열에 지나지 않는다. 이를 분석하여 의미를 추출하기 위해선 의미정보를 담고 있는 온톨로지 컨셉들과 매핑해야 한다. 이를 위해 온톨로지 컨셉은 자연어 표현에 대한 정보를 가지고 있다. 이를 이용해 특정 자연어 문자열이 어떠한 컨셉과 연관되는지 알아낼 수 있다. 이러한 과정을 통해 질의에서 요구하는 서비스의 기능이나 대상, 입출력과 관련된 정보를 추출할 수 있다. 본 절에선 이전 단계에서 분할된 문장블록에 대하여 이와 연관된 서비스 목록을 추출한다.

그림 7과 같이 이전 단계에서 분할된 문장블록에 포함된 동사와 명사 각각에 대하여 action과 object 온톨로지의 컨셉과의 대응 관계를 찾는다. 만약 다수의 object 컨셉이 매핑될 경우 사용자의 의도를 가장 잘 설명하는 하나의 object를 선택한다. 그리고 단일 서비스로 사용자가 요구하는 기능을 충족시키기 힘든 경우가

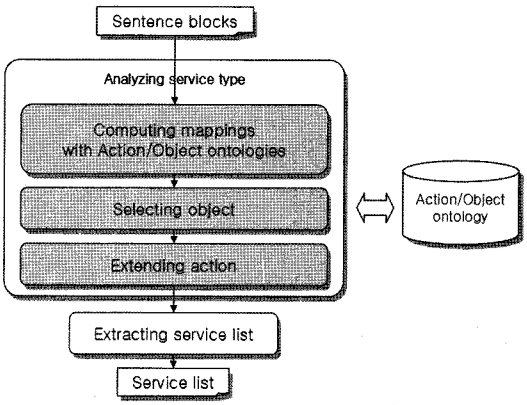


그림 7 문장 블록에 대한 서비스 목록 추출 과정

발생할 수 있다. 이를 대비해 action 컨셉을 확장하여 복합 웹 서비스를 제공한다. action과 object를 이용하여 질의의 유형을 분석한 결과에 따라 단일, 복합 서비스들로 이루어진 서비스 목록을 생성한다.

예를 들어 그림 8 동사 'reserve'는 action 온톨로지의 Reserve 컨셉과 매핑 되고, 명사 'ticket'은 object 온톨로지의 두 개의 컨셉(MassTransit과 Entertainment)과 매핑된다. 질의에서 추출한 명사들을 이용해 매칭한 object컨셉이 많아지면 사용자의 질의에서 요청하는 서비스의 대상이 불명확해질 뿐만 아니라 대상의 범위가 넓어져서 비교해야 할 서비스의 수 역시 늘어난다. 그러므로 이들 컨셉들 중 가장 사용자의 질의에서 요구하는 서비스의 대상을 잘 설명하는 object 컨셉을 선택할 필요가 있다.

즉, 다수의 매핑된 컨셉들을 모두 포괄할 수 있는 컨

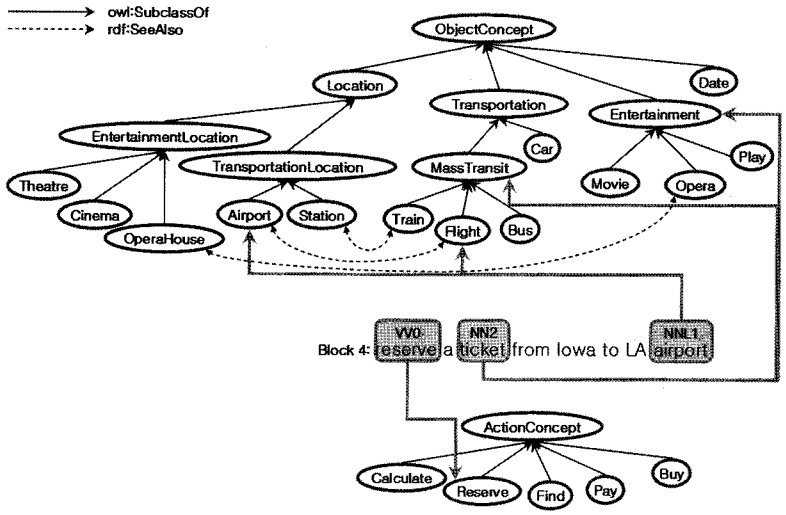


그림 8 키워드로부터 온톨로지 매핑

셉을 찾아 선택한다. 선택하는 과정에서 해당 컨셉이 서로 다른 몇 개의 컨셉들을 포괄하였는지 나타내기 위해 병합 카운터를 제안한다. 선택 과정은 다음과 같다.

- 단계 1: 문장블록 내 키워드들과 매핑된 컨셉들은 모두 선택하고, 해당 컨셉들의 병합 카운터를 1로 설정한다(어떤 컨셉이라도 자기 자신은 포괄할 수 있다).
- 단계 2: 매핑된 컨셉의 참조(rdf:seeAlso) 관계에 있는 컨셉을 선택하고 해당 컨셉의 병합 카운터를 1 증가시킨다. 단, 참조 관계로 새로 선택된 컨셉에 대해 또 다시 참조 관계를 적용하진 않는다(매핑된 컨셉과 유사한 관계에 있는 컨셉들로 확장한다).
- 단계 3: 두 컨셉이 조상(ancestor), 자손(descendant) 관계일 경우 조상 컨셉을 선택 해제하고, 자손 컨셉의 병합 카운터를 조상 컨셉의 병합 카운터만큼 증가시킨다(Object 온톨로지의 자손 컨셉은 조상 컨셉을 보다 구체적으로 기술한다. 자손 컨셉은 조상 컨셉의 특징을 포함하고, 더 많은 특징을 나타낸다).
- 단계 4: 두 컨셉의 공통 조상이 존재할 경우, 공통 조상을 선택하고, 두 자손 컨셉은 선택 해제한다. 공통 조상의 병합 카운터를 두 자손 컨셉의 병합 카운터의 합만큼 증가시킨다. 단, 두 형제 컨셉의 공통 조상이 최상위 컨셉일 경우에는 관계가 없는 것으로 간주한다(두 컨셉의 공통적인 특징을 모두 지닌 조상 컨셉을 선택하여 두 컨셉을 포괄한다).

그림 9는 위의 방법을 적용해 병합 카운터를 사용한 예이다. 먼저 그림의 (a)와 같이 문장 블록과 매핑된 각각의 컨셉의 병합 카운터를 모두 1로 설정한다. 그 다음 참조 관계에 있는 컨셉 Airport와 Flight의 카운터를 각각 1씩 증가시킨다. 그런 다음 부모 자식 관계가 존재하는 컨셉 MassTransit과 Flight 중 Flight을 선택하고, 해당 컨셉의 병합 카운터를 컨셉 MassTransit의 병합 카운터만큼 증가시킨다. 최종적으로 Airport, Flight, Entertainment 세 개의 컨셉이 남게 되며 각각의 컨셉에 대한 병합 카운터의 최종 값은 그림의 (b)에서 나타난 바와 같다.

병합 카운터 값이 클수록 해당 컨셉이 선택 과정에서 질의와 매핑된 컨셉들을 많이 포괄한 것으로 이해할 수 있다. 그림 9의 (b)에 나타난 병합 카운터의 값을 살펴보면 컨셉 Flight이 가장 큰 값을 가진다. 그러므로 Flight 컨셉을 결과로 선택한다. 예외적으로 우선순위의 값이 가장 큰 컨셉이 둘 이상 나타날 경우 해당 컨셉들을 모두 결과로 선택한다.

사용자의 질의에서 동사를 추출하여 action 온톨로지의 컨셉에 매핑하는 과정을 통해 사용자가 원하는 서비스의 기능을 알 수 있다. 그리고 해당 action 컨셉과 매핑된 단일 서비스들에 대해 질의와 유사도를 측정하여

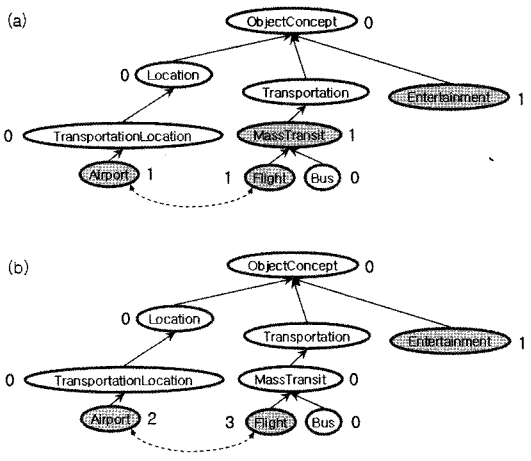
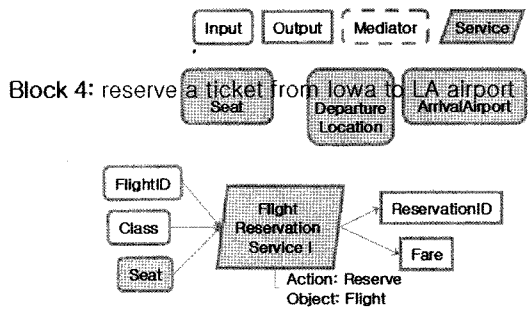


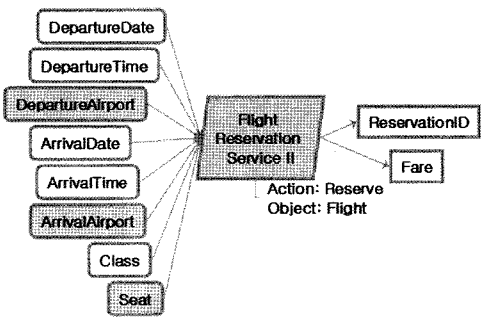
그림 9 Object 선택의 예

적절한 서비스를 찾을 수 있다. 그러나 사용자의 질의를 만족하는 단일 서비스가 존재하지 않을 경우 action 컨셉을 이용한 단일 서비스 탐색 만으로는 사용자의 요구를 해결할 수 없다.

예를 들어 그림 10에서는 사용자가 'reserve'란 동사를 사용해서 항공편을 예약하기 위한 서비스를 요청하고 있다. 사용자는 예매하고자 하는 좌석의 수와 출발하는 장소, 도착하는 공항을 정보로 제공하고 있다. 그러므로



(a) Flight Reservation Service I.



(b) Flight Reservation Service II.

그림 10 단일 서비스 예제

이 세 가지 정보를 받아들이는 '항공편 예약 서비스'를 찾아야 한다. 그림의 (a) 경우처럼 서비스에서 입력으로 요구하는 정보가 사용자가 제공한 정보를 활용하지 못하는 경우가 발생할 수 있다. 또한 (b)의 경우처럼 서비스에서 요구하는 정보의 양이 사용자가 제공한 양 보다 너무 많은 경우가 발생할 수 있다. 제안된 방법은 이러한 경우 그림 11과 같이 서비스를 복합하여 문제를 해결한다. 그림의 (a)의 경우 서비스에서 필요한 입력의 수는 늘었지만 사용자가 제공한 정보를 모두 사용한다. 그림의 (b)의 경우 서비스에서 필요한 입력의 수가 줄어들어 사용자가 추가로 제공해야 할 정보의 수 역시 줄어들었다.

action 컨셉의 확장은 서비스와 온톨로지들의 매핑 시 object 온톨로지에 추가한 정보를 이용한다. object 온톨로지의 각각의 컨셉들은 특정 서비스의 유형을 기술할 때 자신과 같이 쓰인 action 컨셉들의 목록을 가지고 있다. 그림 12는 Flight 컨셉에 기술되어 있는 action 컨셉의 목록(Reserve, Find, Pay)을 보여준다.

사용자가 요청한 '항공권 예매 서비스'를 수행하기 위한 서비스 목록은 그림 10에서 action 컨셉 'Reserve'와 object 컨셉 'Flight'을 이용해 구한 'Flight Reservation Service I, II' 와 같은 단일 서비스들과, action 컨셉을 확장하여 구한 'Flight Search Service I, II'와 'Flight Reservation Service I, II'로 생성한 복합 서비스들로 이루어진다.

3.3 서비스 선택

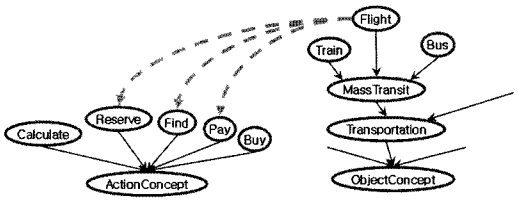
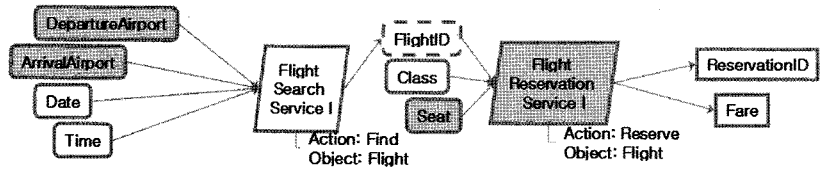


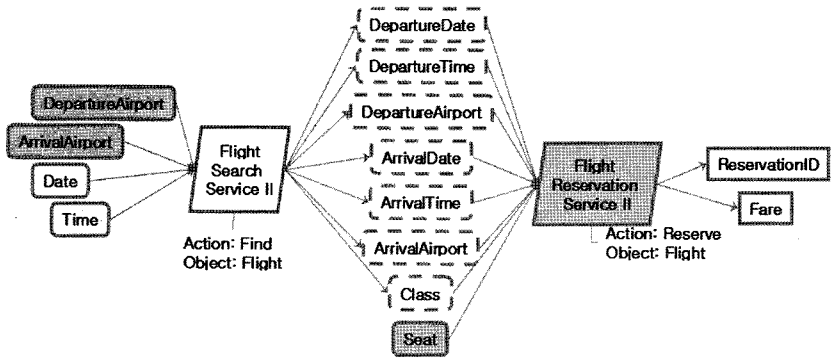
그림 12 Action의 확장

질의에서 요구하는 서비스의 유형을 분석하여 해당 유형의 서비스 목록을 추출하였으면 서비스 목록 중에서 어떠한 서비스가 질의에서 요구하는 서비스인지 선택해야 한다. 제안된 방법은 문장 블록에서 사용된 입출력 값들을 분석하여 서비스와의 유사도 비교를 위해 사용한다. 문장 블록에서 사용된 입출력 값들을 알아내기 위해 입출력 온톨로지의 컨셉들에 포함된 자연어 표현 정보를 이용한다.

그림 13은 서비스 목록에 사용된 입출력 값들을 모두 주어진 예문과 비교하여 문장 블록에서 Seat, DepartureLocation, ArrivalAirport 이렇게 세 가지 입출력 값들을 찾아 서비스에서 사용하는 입출력 값에 매핑하는 방법을 표현한다. 그림에서 Seat과 같이 서비스에서 요구하는 입출력 값과 정확히 매핑된 경우를 ExactMatch라 한다. 그리고 DepartureLocation과 같이 서비스에서 요구하는 입출력 값의 부모 컨셉이 매핑된 경우 PartialMatch라 한다. 또한 FlightID와 Class와 같이 연결된 서비스의 입력으로 사용될 경우 Mediator라 한



(a) Flight Reservation Service I.



(b) Flight Reservation Service II.

그림 11 복합 서비스 예제

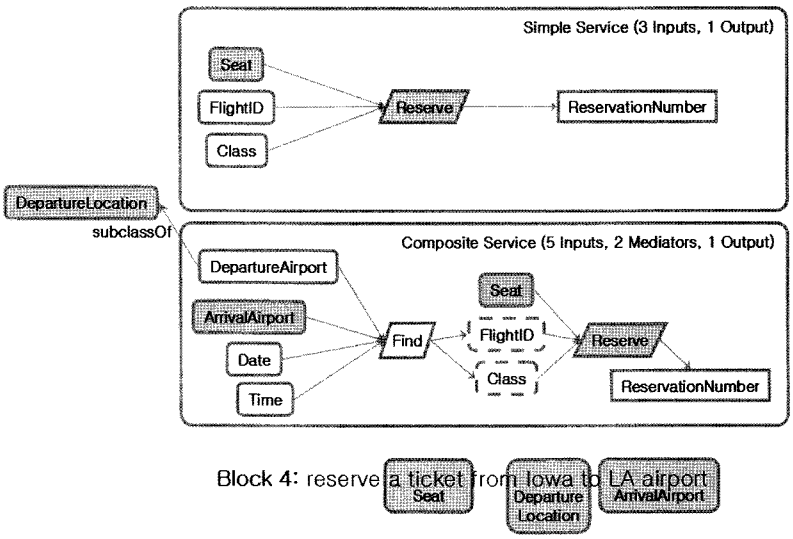


그림 13 문장 블록과 서비스간 입출력 컨셉 비교의 예

다. 문장 블록과 서비스간 유사도 측정은 서비스의 총 입력 컨셉의 수를 TotalInput, 자연어 질의 내에 포함된 컨셉의 수를 OfferedConcept, ExactMatch의 수를 EM, PartialMatch의 수를 PM이라 할 때 다음과 같다. 단 Mediator의 경우, 사용자에게 요구할 필요가 없는 것으로 고려하여 TotalInput에는 포함하지 않는다.

$$ServiceSimilarity = \frac{EM \times 1 + PM \times 0.7}{TotalInput} \times \frac{EM + PM}{OfferedConcept} \quad (1)$$

이 수식은 서비스와 자연어 질의간의 유사도 및 자연어 질의에서 제공한 정보의 활용도를 고려한다. 정보의 활용도를 고려하지 않으면, 단일 입력을 요구하는 서비스의 입력이 ExactMatch일 경우, 사용자가 제공한 정보의 양과 상관 없이 무조건 가장 높은 유사도를 가진다. 또한 ExactMatch와 PartialMatch의 경우를 나누어 차별적인 가중치를 부여하여 ExactMatch의 경우를 보다 우선시 하였다. 해당 가중치는 사전에 실험을 통하여 설정하였다. 그림 13에 이 식을 적용하면 단일 서비스의 경우 유사도가 0.11(EM: 1, PM: 0, TotalInput: 3, OfferedConcept: 3)이 되며 복합 서비스의 경우 0.54 (EM:2, PM: 1, TotalInput: 5, OfferedConcept: 3)가 된다.

3.4 추상 워크플로우의 생성

이전 단계에서 워크플로우 템플릿을 추출하고, 문장 블록마다 유사한 서비스를 선택하였다. 본 단계에선 이 두 가지 정보를 종합하여 추상 워크플로우를 생성한다. 만일 워크플로우 템플릿 추출 단계에서 둘 이상의 템플릿이 추출되었을 경우, 사용자의 의도에 가장 적합한 워크플로우를 찾아야 한다. 제안된 방법은 워크플로우가 사용자의 의도에 적합한지 측정하기 위해 워크플로우를 구성하는 하위 워크플로우(sub workflow) 간의 의미적 및 구조적 균형을 고려하여 적절한 워크플로우를 선택한다.

예를 들어, 그림 14는 'and'와 'or' 제어 구문이 모두 포함된 경우, 추출 가능한 워크플로우 템플릿을 보여준다. 'and'와 'or'를 결합하는 방법에 따라 그림에 나타난 것과 같이 두 가지로 해석이 가능하다. 그러나 Block4와 Block5가 서로 유사한 서비스(숙박 시설 예약)를 요청한 것으로 비추어 볼 때, 사용자의 의도는 Block3을 먼저 실행하고, Block4와 Block5중 하나를 선택하여 수행하길 원하는 것으로 해석 가능하다.

그림 15에서 든 예는 세 개의 제어 구문을 사용하고 있어서 그림 14보다 많은 후보 템플릿이 생성 가능하다. 그러나 구조적인 균형을 고려하면 Block1(열차표 예약)과 Block2(호텔 예약)를 수행하거나 Block3(항공권 예약)과 Block4(게스트 하우스 예약)를 수행하는 것으로 해석할 수 있다. 제안된 방법은 'and'나 'or' 구문으로 연결된 하위 워크플로우 그룹간의 의미적 및 구조적 균

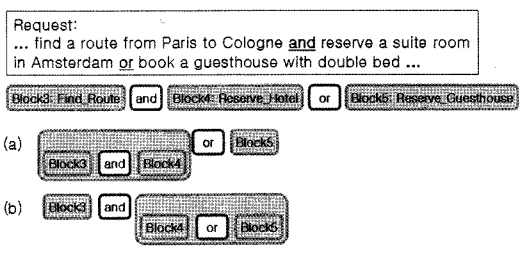


그림 14 의미적 균형을 통한 워크플로우 선택

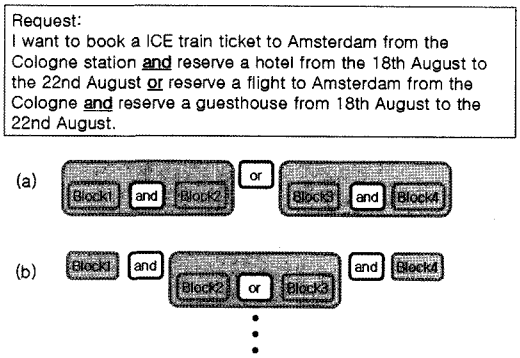


그림 15 구조적 균형을 통한 워크플로우 선택

형을 고려하여 사용자의 의도에 맞는 워크플로우를 선택한다. 제안된 방법은 'and'나 'or' 제어 구문 전후의 하위 워크플로우들을 비교하여 의미적 또는 구조적으로 서로 유사한지 비교한다.

단일의 'if-then-else' 제어 구문에 속한 'then'절과 'else'절 혹은 'if'절과 'else'절은 의미적 또는 구조적으로 유사하게 사용되는 경우가 많다. 이러한 점을 이용하여 'if-then'과 'if-then-else'가 중첩하여 등장할 경우 'if'와 'else'절 그리고 'then'과 'else'절의 하위 워크플로우를 비교한다.

$$SubWorkflowSimilarity = StructureSimilarity + ActionConceptSimilarity + ObjectConceptSimilarity \quad (2)$$

$$\{Action|Object\}ConceptSimilarity = \frac{EM \times 1 + PM \times 0.7}{NumberOfConcepts} \quad (3)$$

하위 워크플로우 간의 유사도는 식 (2)와 (3)으로 계산한다. 비교 대상인 두 하위 워크플로우에 속하는 문장 블록들과 매핑된 object 컨셉들의 수를 NumberOfConcepts라 한다. 그리고 두 하위 워크플로우의 object 컨셉 중 정확히 일치하는 컨셉들의 수를 EM, 정확히 일치하진 않으나 부모 자식 관계나 형제 관계인 컨셉들의 수를 PM라 한다. 여기서 해당 가중치는 식 (1)과 마찬가지로 사전에 실험을 통하여 설정하였다. 그리고 두 하위 워크플로우가 구조적으로 동일할 경우, 구조 유사도를 1로 설정하고, 그렇지 않을 경우에는 0으로 설정한다.

그림 16은 그림 6에서 예로 사용한 요청으로부터 추출 가능한 다수의 워크플로우 중 두 가지를 보여준다. 각각의 계산된 값을 합하여 워크플로우의 균형을 측정하는 지표로 사용하며, 가장 큰 값을 갖는 워크플로우를 선택한다. 그림 16의 왼쪽 워크플로우의 경우, 'if-then-else'에 중첩된 'then'과 'else' 구문은 동일한 구조와 action 컨셉을 갖는다. 또한 3개의 object 컨셉 중에서 2개는 서로 일치한다. 따라서 두 하위 워크플로우 간의 유사도는 2.67(StructureSimilarity: 1, ActionConceptSimilarity: 1, ObjectConceptSimilarity: 0.67)로 계산된다.

제안된 방법은 선택된 워크플로우 템플릿에 이전 단계에서 각각의 문장 블록에 대해 찾은 서비스 정보를 결합하여 최종 추상 워크플로우를 생성한다. 그림 17은 그림 1에서 예로 든 요청에 대해 추상 워크플로우를 생성한 결과를 나타낸다.

4. 실험 결과

본 절에서는 제안된 방법을 평가하기 위해 시스템을 구현하여 제안된 방법의 효용성을 알아본다. 실험은 127개의 질의를 사용하였으며 웹에서 수집한 50개의 간단한 질의들과 수집한 질의를 조합 및 가공하여 77개의 복잡한 질의를 추가로 생성하였다. 실험에 사용한 질의들의 복잡도에 따른 분류는 표 1과 같다. 그리고 웹에서 추출한 질의들을 도메인 별로 분석한 내용이 표 2에 나타나 있다.

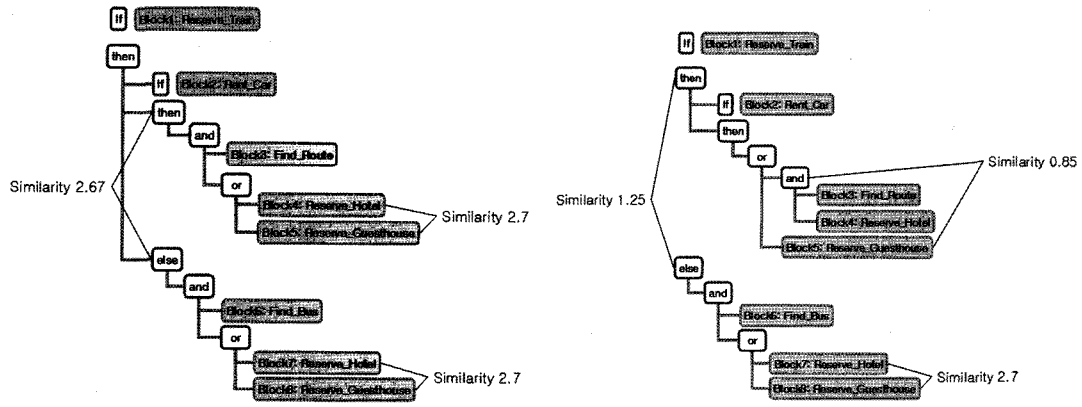


그림 16 워크플로우 유사도의 계산

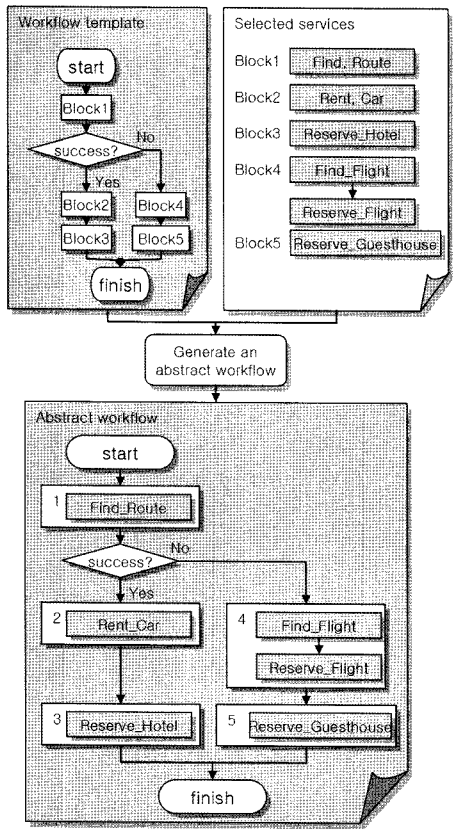


그림 17 추상 워크플로우의 생성

표 1 복잡도에 따른 질의 분류

Complexity	# of Queries
Low	50
Medium	46
High	31
Total	127

질의의 복잡도에 따라 low, medium, high로 나누었으며 low는 제어 구문이 없고, 1~2개의 절로 구성되었다. medium은 1개의 제어 구문을 포함하며 2개 이상의 절을 포함하고 있다. high의 경우 2개 이상의 제어 구문을 포함하며 3개 이상의 절들로 구성되어 있다. 표 3은 실험에 사용한 질의 내에 포함된 제어 구문의 총 수를 나타낸다.

4.1 성능 분석

전문가가 수작업으로 질의를 분석해 생성한 추상 워크플로우와 제안한 방법을 이용해 자동으로 생성한 추상 워크플로우를 비교한 결과는 표 4와 같다.

추상 워크플로우를 잘못 생성한 경우를 분석해 보면 두 가지로 나눌 수 있다. 첫 번째는 문장 블록에 대해

표 2 수집한 질의의 도메인 분포

Domain \ Source	Yahoo	Google
Accommodation	1	4
Book	1	5
Camera	6	1
Car	1	
Computer	3	1
DVD	3	
Mass Transit	6	
Movie	4	
MP3Player	1	
Music	3	
Photo	2	
Route	3	
Shipping	3	
Traffic Information	1	
Translation	1	
Total	39	11

표 3 질의에 포함된 제어 구문의 수

	Low	Medium	High	Total
if-then-else	0	7	17	24
sequence	0	9	34	43
choice	0	8	9	17
split-join	0	9	9	18
repeat	0	6	7	13
any order	0	7	7	14
Total	0	46	83	129

표 4 실험 결과

	Service mismatch	Incorrect template	Success rate
Low	2	0	96%
Medium	1	0	97.8%
High	2	1	90.3%
Total	5	1	95.2%

잘못된 서비스를 매칭하는 경우이다. 하나의 문장 블록을 서비스와 매치하기 위해 문장 블록에서 요구되는 서비스의 유형을 분석해야 한다. 그러나 분석이 실패할 경우 서비스와 매치가 불가능하다. 제안한 object 온톨로지의 컨셉의 자연어 표현 정보는 주로 주제가 있는 단어 (thematic keyword) 들로 구성되어 있다. 예를 들어 Car컨셉의 경우 "car"나 "vehicle", "sedan"과 같은 단어들로 표현한다. 그러나 "rent a Honda accord."와 같은 문장 블록이 들어올 경우 이 문장을 Car컨셉에 대한 것으로 판별해내지 못한다.

오류가 발생하는 다른 한가지는 사용자가 요청한 질의에서 의도하는 추상 워크플로우의 모습과 다른 워크

플로우를 추출한 경우이다. 다음 두 가지 예문을 이용해 설명해 보겠다.

"I want to reserve a train ticket to California from Iowa or reserve a flight ticket from Iowa to California and reserve a hotel suite room in California from 23rd August to 25th August."

"I want to rent a car which has an air bag in Iowa or reserve a flight ticket from Iowa to California and reserve a hotel suite room in California from 23rd August to 25th August."

그림 18은 두 가지 예문에 대해 생성한 워크플로우를 나타낸다. Request1은 첫 번째 요청에 대해 제안한 방법으로 해석한 결과이다. 이 결과는 사용자의 의도에도 부합하고, 질의 자체에 모호함도 거의 없다. 그러나 두 번째 예문의 결과인 Request 2-1과 Request 2-2의 경우 사용자의 의도는 비교적 Request 2-1에 가까운 반면 제안한 방법은 Request 2-2를 선택하게 된다. 그 원인은 다음과 같다. Rent_Car와 Reserve_Flight간의 유사도는 상당히 낮다. 그러나 Reserve_Flight과 Reserve_Hotel의 경우 'reserve' 동사가 동일하게 사용되어 앞선 비교보다 더 높은 유사도를 가진다.

본 논문에서 추상 워크플로우를 추출하기 위해 사용자의 요청으로부터 모든 가능한 워크플로우의 형태를 이끌어 낸 다음 의미적, 구조적 분석을 통해 사용자의 요청에 가장 적합한 워크플로우의 형태를 선택하는 방법을 제안하였다. 이러한 방법에 따라 사용자의 요청에서 상당히 많은 워크플로우의 형태가 추출되게 된다. 예를 들어 그림 19와 같이 'and'와 'or'같은 이진 제어구문이 세 개 사용된 문장이 있을 경우 추출 가능한 워크플로우의 가지 수는 다섯 가지이다.

이진 제어 구문만 사용된 문장의 경우 추출 가능한 워크플로우의 가지 수는 문장 내에서 사용된 이진 제어 구문을 순차적으로 추출하는 방법의 가지 수와 같다. 물론 중복된 워크플로우가 다수 생성되나 이에 대해 고려하지 않고 총 워크플로우의 수를 계산하면 순열을 구하는 공식을 사용하여 구할 수 있다.

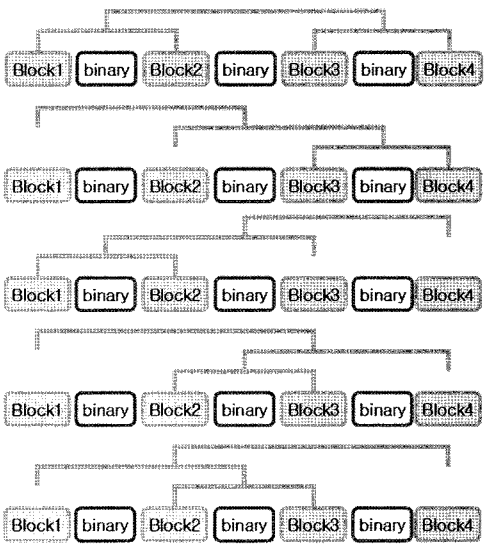


그림 19 이진 제어구문의 다양한 해석

$$Number\ Of\ Extracted\ Workflows = {}_nP_r = \frac{n!}{(n-r)!} = n! \quad (\because n = r)$$

이 계산식에서 n은 문장 내의 이진 제어 구문의 수를 뜻하며 r은 문장 내에서 뽑을 제어 구문의 수를 나타낸다. 문장 내의 모든 제어구문을 추출해야 하므로 r은 n과 같다. 이 식을 그림 19에 적용하면 n과 r은 3이고, 결과적으로 총 워크플로우의 수는 6이 된다. 이 중 한 가지 워크플로우가 중복되어 실제로 다섯 가지의 워크플로우가 추출된다.

워크플로우를 추출하기 위한 복잡도는 하나의 문장에서 추출 가능한 모든 워크플로우를 추출하는 과정에 비례한다. 그러므로 문장에서 추출 가능한 워크플로우의

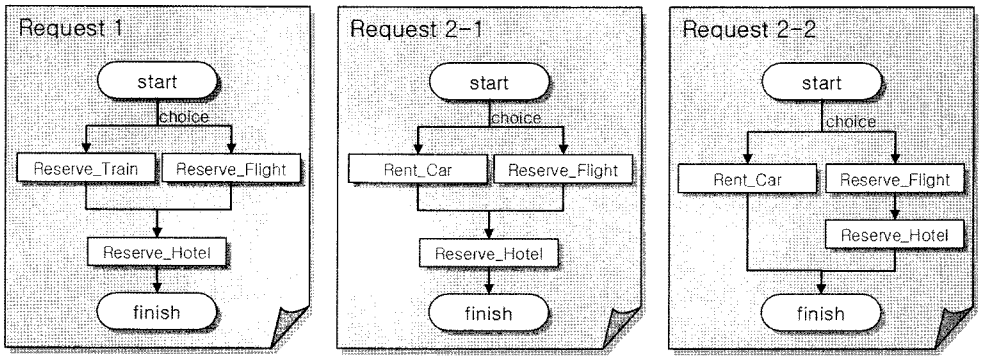


그림 18 잘못된 워크플로우 생성 예

수를 구하는 식을 이용하여 복잡도를 구하면 다음과 같다.

$$Complexity : O(n) = n!$$

이러한 복잡도는 n이 커짐에 따라 매우 급격하게 증가하게 된다. 그러므로 본 논문에서 제안한 방법은 지나치게 많은 제어 구문을 포함한 문장으로부터 워크플로우를 뽑아내기엔 어려움이 있다. 그러나 'if-then-else'가 포함된 문장은 해당 제어구문이 문장을 'if'와 'then' 그리고 'else' 구절로 나누어서 처리하기 때문에 복잡도에서 n의 크기를 여러 개로 나누어서 처리하는 효과를 볼 수 있다. 그리고 사용자의 요청을 여러 문장으로 나누어서 표현할 경우 좀 더 많은 제어 구문이 포함된 요청을 처리 가능하다. 그리고 현실적으로 만은 제어구문을 포함한 문장을 사용자들이 작성하는 것 자체도 역시 쉽지 않다.

4.2 기존 연구와의 비교

본 절에서는 제안한 방법과 기존 연구의 내용을 정성적으로 비교 분석한 내용에 대해 기술한다. 표 5에서 보여지는 바와 같이 기존 연구는 하나의 문장에서 간단한 워크플로우만 추출 가능하며 제어 구문이 한 문장 내에 중복적으로 등장하여 복잡한 로직을 구성할 경우 처리하지 못한다.

기존 연구 중 Bosca 등은 한 문장에서 'if-then'이나 'sequence'와 같이 하나의 워크플로우만 추출한다. 그리고 Xie 등은 문장 내에서는 워크플로우를 추출하지 못하고, 복수의 문장을 연결하는 방법으로 워크플로우를 지원한다. Englmeier 등 이 제안한 방법 역시 문장 내 워크플로우를 추출하는 방식이 아닌 간단한 문장들과 제어 구문을 나타내는 통제된(Controlled) 언어를 사용하여 문장간의 제어 구조를 이끌어 낸다.

본 논문에서 제안한 방법은 아래의 예문과 같이 하나의 문장 내에 여러 서비스와 제어 구문이 복합적으로 표현된 요청을 처리할 수 있다. 제안된 방법은 한 문장 내에서 제한 없이 매우 많은 서비스들을 표현 할 수 있으나 워크플로우를 생성하는 과정에서 성능상의 문제가 발생할 순 있다.

"**If** I can find a Highway from Iowa to California **then** rent a car in Iowa **and** reserve a hotel suite room from 23rd August to 25th August in California **else** reserve a ticket from Iowa to LA airport **and** reserve a guesthouse with twin bed from 18th August to 22nd August."

기존 연구들은 대체적으로 문장 블록에 대해 하나의 서비스를 매핑한다. 그러나 본 논문에서는 단일 서비스와 사용자가 요청한 서비스의 기능(Action)을 확장하여 구한 복합 서비스 중 사용자의 의도를 가장 잘 반영한 것을 선택하는 방법을 제안하였다. 이를 통해 단일 서비스로 사용자의 요청을 적절하게 수행할 수 없는 경우에도 복합 서비스를 제공하여 사용자의 요청을 만족시킬 수 있다. 물론 Xie 등이 제안한 방법에서 Object를 확장하여 복합 서비스를 매칭하는 경우가 있다. 그러나 이 방법은 사용자가 '여행 일정'에 대한 서비스를 요청했을 경우 이를 확장하여 '항공편 예약', '숙박 예약' 그리고 '자동차 대여'에 대한 서비스 요청과 동일시 하는 방법이다. 이러한 방법은 각각의 '항공편 예약', '숙박 예약' 과 같은 확장된 서비스에 대해 마찬가지로 매칭 가능한 단일 서비스가 없을 경우 사용자의 요청을 만족시킬 수 없다.

그림 20은 간단한 예문에 대해 기존 연구 중 Bosca 등 의 방법과 본 논문에서 제안한 방법의 차이를 보여준다. 이 경우 만약 'Publisher'를 받아들일 수 있는 'Buy' 서비스가 존재할 경우에는 Bosca 등 의 방법도 사용자의 질의를 만족시킬 수 있으나 그렇지 못 할 경우 본 논문에서 제안한 'Buy'서비스와 연관 있는 서비스들을 최대한 이용하는 방법이 보다 사용자의 질의를 만족시킨다.

5. 결 론

웹 서비스가 다양한 분야에 넓게 확산됨에 따라 복합 웹 서비스 구성에 대한 관심 또한 증가하고 있다. 그러나 일반 사용자들이 직접 복합 웹 서비스 구성하는 데에는 상당한 어려움이 따른다. 그리하여 복합 웹 서비스

표 5 제안한 방법과 기존 연구 비교

기준*	Al-Muhammed	Bosca	Englmeier	Jang	Xie	제안된 방법
1	X	X	controlled	X	X	X
2	X	△ (single logic)	X	X	X	O
3	X	X	X	X	O	O
4	O	O	O	O	O	O
5	O	O	O	O	O	O
6	O	X	O	X	O	O
7	X	X	X	X	O	O

*1: 입력문법 지정여부, 2: 워크플로우 추출여부, 3: 단일 문장블럭으로부터 복합서비스 추출, 4: 매개변수 추출, 5: 지식 활용, 6: 복수 문장블럭 처리(단일 도메인), 7: 복수 문장블럭 처리(다중 도메인)

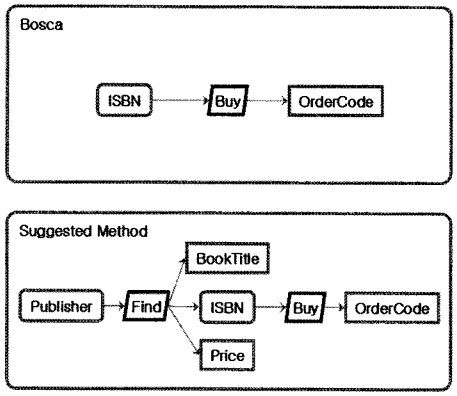


그림 20 기존 연구와의 차이점 비교

를 편리하게 생성하고, 호출하는 인터페이스가 필요하다. 이에 대해 자연어 질의를 이용한 복합 웹 서비스 생성 인터페이스에 대한 연구들이 제안되었다. 대부분의 기존 연구들이 간단한 제어 구문에 대해서만 처리 하거나, 하나의 문장 블록에 대해 단일 서비스만을 대응한다. 본 논문에서는 사용자의 자연어 질의로부터 정교한 수준의 추상 워크플로우를 생성하는 방법을 제안한다. 복잡한 제어 구문들을 처리하기 위해 제안된 방법은 간단한 제어 구문의 패턴을 반복적으로 적용하여 복잡한 워크플로우를 추출한다. 그리고 각각의 문장 블록에 포함된 명사와 동사를 이용하여 action과 object를 찾는다. Action과 object는 문장 블록에서 요구하는 서비스의 기능과 대상을 표현하며, 문장 블록에서 요구하는 서비스를 찾기 위해 사용한다. 찾은 서비스들은 추출한 워크플로우의 태스크에 대응되어 최종 추상 워크플로우를 완성한다. 제안된 방법의 성능을 평가하기 위해 다양한 사용자 질의를 이용하여 실험을 진행하였다. 제안된 방법은 95.2%의 질의에 대해 정교한 추상 워크플로우를 정확히 생성하였다.

참 고 문 헌

[1] World Wide Web Consortium, Web Services, http://www.w3c.org/2005/ws.

[2] Bosca, A., Valetto, G., Maglione, R., and Corno, F., "Specifying Web Service Compositions on the Basis of Natural Language Requests," *Proc. International Conference on Service Oriented Computing*, pp.588-593, 2005.

[3] Bosca, A., Ferrato, A., Corno, F., Congju, I., and Valetto, G., "Composing Web Services on the Basis of Natural Language Requests," *Proc. IEEE International Conference on Web Services*, pp. 817-818, 2005.

[4] Bosca, A., Corno, F., Valetto, G., and Maglione, R., "On-the-fly Construction of Web Services Compositions from Natural Language Requests," *Journal of Software*, vol.1, no.1, pp.40-50, 2006.

[5] Xie, F., Gong, H., Deng, D., Wang, S., Wang, G. T., Hu, J., and Sheu, P. C. Y., "Integrating Semantic Web Services for Declarative Accesses in Natural Language," *Proc. IEEE International Symposium on Multimedia*, pp.201-208, 2006.

[6] Englmeier, K., Pereira, J., and Mothe, J., "Choreography of web services based on natural language storybooks," *Proc. International Conference on Electronic Commerce*, pp.132-138, 2006.

[7] Jang, M., Sohn, J. C., and Cho, H. K., "Automated Question Answering Using Semantic Web Services," *Proc. IEEE Asia-Pacific Service Computing Conference*, pp.344-348, 2007.

[8] Al-Muhammed, M. J. and Embley, D. W., "Conceptual Model Based Semantic Web Services," *Proc. International Conference on Conceptual Modeling, LNCS 3716*, pp.288-303, 2005.

[9] Al-Muhammed, M. J. and Embley, D. W., "Resolving Underconstrained and Overconstrained Systems of Conjunctive Constraints for Service Requests," *Proc. International Conference on Advanced Information Systems Engineering*, pp.223-238, 2006.

[10] Al-Muhammed, M. J. and Embley, D. W., "Ontology-Based Constraint Recognition for Free-Form Service Requests," *Proc. IEEE International Conference on Data Engineering*, pp.366-375, 2007.

[11] Semantic Markup for Web Services (OWL-S), http://www.w3.org/submission/owl-s.

[12] Briscoe, T., Carroll, J., and Watson, R., "The Second Release of the RASP System," *Proc. the COLING/ACL Conference*, pp.77-80, 2006.

임 종 현

2006년 연세대학교 컴퓨터과학과 졸업(학사). 2008년 연세대학교 컴퓨터과학과 졸업(석사). 2008년~현재 LG전자 CTO DTV연구소 연구원. 관심분야는 Service-Oriented Computing, Web Services Composition

이 경 호

1995년 연세대학교 전산과학과 졸업(학사). 1997년 연세대학교 컴퓨터과학과 졸업(석사). 2001년 연세대학교 컴퓨터과학과 졸업(박사). 2001년~2002년 National Institute of Standard and Technology (NIST) 객원연구원. 2002년~현재 연세대학교 컴퓨터과학과 부교수. 관심분야는 Service-Oriented Computing, Multimedia/Web Engineering, Semantic Web