상호주도형 대화 에이전트 구현을 위한 도메인 독립적 스크립트 언어

임성수° 조성배
연세대학교 컴퓨터과학과
lss@sclab.yonsei.ac.kr°, sbcho@cs.yonsei.ac.kr

Domain Independent Script Language for Constructing Mixed-Initiative Conversational Agent

Sungsoo Lim° Sung-Bae Cho
Dept. of Computer Science, Yonsei University

요 약

대화 에이전트의 역할은 사용자 입력으로부터 사용자의 의도를 분석하고 이에 따른 서비스를 제공하는 것이다. 하지만 사용자는 한 번에 서비스 제공에 필요한 모든 정보를 제공하지 않으므로 에이전트는 능동적으로 부가적인 정보를 추출할 수 있어야 한다. 따라서 자연스러운 대화를 진행하기 위해서 에이전트는 사용자주도형 대화와 시스템주도형 대화가 결합된 상호주도형 대화가 가능해야 한다. 본 논문에서는 상호주도형 대화를 제공할 수 있는 대화 에이전트의 스크립트(대화를 위한 데이터베이스) 언어를 제안한다. 제안한 방법은 대상 도메인에 맞춰서 대화 에이전트를 설계할 수 있도록, 도메인 관련 변수와 도메인 함수를 정의하여 사용할 수 있으며, 대화처리 기능으로, 사용자 의도 추론, 대화 흐름 관리, 사용자 입력 정보 추출 등의 기능을 지원한다. 제안한 방법의 가능성을 보이기 위해 일정관리 도메인에 제안하는 방법을 적용한다.

1. 서론

인간과 컴퓨터의 상호작용은 사람이 컴퓨터를 이용하여 지능적인 행동을 수행할 수 있도록 도와준다. 인간이 서로 의사소통 수단으로 사용하는 자연어는 유연성, 명료성, 표현력 면에서 뛰어난 장점을 가지고 있다. 이러한 자연어를 인간-컴퓨터간의 통신에 사용한다면, 기존의 컴퓨터 제약적이거나 사용자 제약적인 시스템에서 제공하는 메뉴와 같은 정보전달방식과는 달리, 상호작용을 풍부하게 하고 단순히 하나의 단어나 사용자 입력을 통해서 전달하는 정보에 비해 훨씬 복잡한 정보를 포함할 수 있다[1,2].

이러한 자연언어의 장점을 효과적으로 활용하기 위해 최근 자연어로 해당 분야의 전문 지식을 알려주는 대화 에이전트의 연구가 활발히 진행되고 있다. 대화 에이전트의 역할은 사용자 입력으로부터 사용자의 의도를 분석하고 이에 따른 서비스를 제공하는 것이다. 하지만 사용자는 한 번에 서비스 제공에 필요한 모든 정보를 제공하지 않으므로 에이전트는 능동적으로 부가적인 정보를 추출할 수 있어야 한다. 따라서 자연스러운 대화를 진행하기 위해서 에이전트는 사용자주도형 대화와 시스템주도형 대화가 결합된 상호주도형 대화가 가능해야 한다.

시스템주도형 대화는 미리 정해진 흐름대로 시스템이 물어보고 사용자는 그에 대한 대답을 하는 방식으로 대화가 진행되어, 대화의 흐름에서 벗어난 입력을 허용하지 않는다. 반대로 사용자주도형 대화는 대화의 유연성을 제공하지만 사용자가 시스템이 제공해주는 서비스의 범위를 알기 어렵다는 단점이 있다[3].

상호주도형 대화는 시스템주도형 대화와 사용자주도형 대화의 단점을 보완하면서, 장점을 부각시키는 방법이나, 그 설계가 어렵다는 단점이 있다.

초기의 상호주도형 대화는 설계자가 모든 대화의 흐름을 모델링하여 구현되었으나, 이 방법은 많은 시간과 노력이 필요하므로 최근에는 대화의 흐름을 통계적 기법을 이용하여 학습하는 방법들이 제안되었다[4,5,6]. 이러한 통계적 기법은 설계시 필요한 시간과 노력을 줄일 수는 있으나, 학습을 위해서 충분한 양의 데이터가 필요하다.

이와 다른 방법으로는 베이지안 네트워크를 이용한 방법이 있다[3,7]. 베이지안 네트워크를 이용하여 대화를 모델링하면, 대화의 흐름을 확률적으로 추론하면서 유연한 대화를 진행할 수 있지만, 베이지안 네트워크의 설계는 초보자들이 수행하기 어렵고, 설계에 있어서 많은 시간과 노력이 필요하다.

본 논문에서는 초보자들도 손쉽게 상호주도형 대화 에이전트를 설계할 수 있도록, 상호주도형 대화를 제공할 수 있는 대화 에이전트의 스크립트(대화를 위한 데이터베이스) 언어를 제안한다. 제안한 방법은 대상 도메인에 맞춰서 대화 에이전트를 설계할 수 있도록, 도메인 관련 변수와 도메인 함수를 정의하여 사용할 수 있으며, 대화처리 기능으로, 사용자 의도 추론, 대화 흐름 관리, 사용자 입력정보 추출 등의 기능을 지원한다.

2. 대화 에이전트의 지식 표현

대화 에이전트는 사용자 입력에 대한 서비스를 제공하기 위하여 이에 필요한 행동 규칙을 스크립트에 기술한다. 표 1은 대화 에이전트에 사용되는 지식 표현 방법을 보여준다.

표 1의 다양한 기법들을 토대로 대화 에이전트는 웹페이지 소개 도우미나 프로그램 사용 도우미[8], 상품구매 도우미, 여행경로 안내 도우미[9] 등으로 개발되었다. 이외에도 상용화된 대화 에이전트로는 NativeMinds사의 Nicole(www.nativeminds.com), Artificial Life사의 SmartBot(www.artificiallife.com), Virtual Personalities사의 Verbot(www.vperson.com), 국내에서 개발된 MSN messenger상의 심심이, 휴보그(www.huborg.com) 등이 있다.

본 논문에서는 대화프레임기반모델과 계획기반모델을 결합하여 상호주도형 대화를 지원하는 하이브리드 모델을 제안하고 이를 위한 스크립트 언어를 설계 및 구현한다.

표 1. 대화 에이전트에 사용되는 방법

기술	수행 작업 예	기술난이도	
단순패턴매칭	웹검색엔진	단순질의응답	쉬움
제한대본기반모델	게임	단순한 선택을 통한 대화진행	
유한상태모델	장거리전화	상태전환을 통한 제한된 유연성	
프레임기반모델	열차정보검색	단순패턴매칭을 이용한 정보기입	
계획기반모델	부엌디자인도우미	동적주제변화	복잡함

3. 제안하는 방법

본 장에서는 제안하는 스크립트 언어의 구조와 그 동작에 대해서 설명한다. 제안하는 스크립트 언어에서는 대화 에이전트의 단순 질의응답 뿐만 아니라 대화 수행을 위해 필요한 정보를 에이전트가 직접 수집하는 상호주도형 대화도 가능하도록 설계한다. 표 2와 3은 제안하는 스크립트 언어와 여기서 사용된 도메인 관련 변수를 표현하는 템플릿 변수를 BNF 형식으로 보여주고 있다. 제안하는 언어는 초보자도 쉽게 익힐 수 있도록 AIML 형식으로 되어 있으며, 각 태그의 역할은 다음과 같다.

(1) 스크립트 언어

- script: 에이전트가 수행해야할 서비스 단위나 대화의 범주에 따라 하나의 스크립트를 생성한다.
- topic: 해당 스크립트가 속한 주제를 명시한다. 명시된 주제는 대화의 원활한 흐름을 관리하기 위해 주제 스택과 주제 큐에 각각 기록된다.
- pre_topic: 해당 스크립트가 실행되기 위한 이전상태에 대한 명시로, 스크립트 선택시 일치하는 주제 큐에 해당하는 주제가 존재하면 스크립트에 가산점이 부여된다.
- necessary_var: 스크립트 행동(action 태그) 수행에 있어서 필수적으로 필요한 정보를 기술한다. 해당 정보가 없으면, 정보를 얻기 위한 시스템주도형 대화가 시작된다.
- candidate_var: 추천과 같은 상황에서, 반드시 필요하지는 않으나 행동 수행에 있어서 있으면 도움이 될만한 변수에 대한 명시로, extract_candidate_var 태그에 의해서 해당 정보를 얻기 위한 시스템주도형 대화

표 2. 스크립트 언어

```
[topic_name] := @String
[var_string] := @String
[query_string] := @String
[answer_string] := @String
[function_name] := @String
[return_value] := @String

[topic] := <topic> [topic_name] </topic>
[pre_topic] := <pre_topic> [topic_name]+ </pre_topic>
[variable] := <variable] [var_string] </variable>
[necessary_var] :=
        <necessary_var> [variable]+ </necessary_var>
[candidate_var] :=
         <candidate_var> [variable]+ </candidate_var>
[query] := <query> [query_string] </query>
[internal_function] :=
          <change_topic> [topic_name] </change_topic>
        | <sub_topic> [topic_name] </sub_topic]
        | <end_topic/>
        | <extract_candidate_var/>
        | <answer> [answer_string] </answer>
[parameter] = [var_string] | &[var_string]
[function_form] := [function_name]({|[parameter]}+)
[return] :=
    <return value = [return_value]> [order]+ </return>
[function_body] := [function_form] [return]+
[extern_function] :=
                  <function> [function_body] </function>
[order] := [internal_function] | [extern_function]
[action] := <action> [order]+ </action>
[topic_question_body] := [question_string] | [order]
[topic_question] := <topic_question>
                    [topic_question_body] </topic_question>
[script_body] := [topic] ([pre_topic]) ([necessary_var])
([candidate_var]) [query]+ [action]+ [topic_question]+
[script] := <script> [script_body] </script>
```

표 3. 템플릿 변수

```
[object_name] := @String
[var_string] := @String
[default_value] := @String
[value_string] := @String
[question_string] := @String

[object] := <object> [object_name] </object>
[variable] := <variable> [var_string] </variable>
[default] := <default> [default_value] </default>
[value] := <value> [value_string] </value>
[question] := <question> </question>
[tpl_question_body] := [question_string] | [order]
[tpl_question] :=
      <tpl_question> [tpl_question_body] </tpl_question>
[template_body] := [object] [variable] ([default])
                   [value]+ [tpl_question]+
[template] := <template> [template_body] </template>
```

가 시작된다.

- query: 스크립트 선택과 정보 추출을 위해 사용되는 태그로, 사용자 입력의 패턴을 명시한다.
- action: 스크립트의 행동을 명시하는 태그로, 에이전트는 여러 행동중 임으로 하나를 선택하여 행동하고 그 안에 있는 [order]를 순서대로 수행한다.
- change_topic: 상황에 따라서 현재 진행중인 대화를 변경해야하는 경우가 발생한다. 이러한 경우에 주제 스택의 내용을 변경시켜 대화의 흐름을 원활히 할 수 있어야 한다. change_topic 태그는 주제 스택의 top에 위치한 주제를 change_topic 태그 사이에 명시된 주제로 변경한다.
- sub_topic: change_topic과 같은 목적으로 정의된 태그로, 주제 스택에 새로운 주제를 푸쉬(push)한다.
- end_topic: 해당 스크립트의 목적을 모두 완료되었음을 에이전트에게 알려주는 태그로, end_topic 태그를 만나면 에이전트는 주제 스택에서 해당 주제를 팝(pop)한다.
- answer: [answer_string]을 답변으로 출력하며, [answer_string]에 템플릿 변수가 존재하면 해당 변수의 값을 변수 명과 대체하여 출력한다.
- function: 해당 도메인에 관련된 기능을 수행할 수 있도록 제공되는 태그이다. 사용할 수 있는 함수는 리턴 값으로 string을 가지며, 파라메터로는 string이나 string*로 제한한다. 사용자 지정함수의 수행결과에 따라 다른 행동을 수행할 수 있도록 return 태그에 대해 정의한다.
- topic_question: 비슷한 점수를 가지는 복수의 스크립트가 선택되었을 때, 사용자의 의도를 보다 정확하게 파악하기 위해 에이전트에게 필요한 질의문을 명시한다.

(2) 템플릿 변수

- object: '일정,' '주소록' 등과 같은 개체의 범주에 대해 명시한다.
- variable: 추출할 변수에 대한 명시로, 필요하면 그 추출 패턴과 함께 입력한다.
- value: 해당 변수가 가질 수 있는 값을 명시한다.
- default: 스크립트의 answer 태그에서 템플릿 변수 값이 존재하지 않을 때 사용될 값에 대해 명시한다.
- tpl_question: 해당 변수 값을 사용자에게 물어보기 위한 방법을 명시한다.

제안하는 스크립트 언어에서는 대화 에이전트 설계자가 도메인에 필요한 템플릿 변수를 직접 명시할 수 있고, 도메인에 필요한 기능을 수행할 수 있도록 사용자 정의 함수를 동작시킬 수 있도록 설계되었다. 이러한 설계는 제안하는 스크립트 언어를 통해 다양한 도메인에서의 대화 에이전트를 설계할 수 있도록 해서 에이전트의 확장성을 높여준다.

3.1 스크립트의 선택

스크립트의 선택은 기본적으로 사용자 입력문과 스크립트의 질의문과의 순차패턴매칭을 통해서 결정된다. 순차 패턴매칭의 평가점수는 문서분류에서 많이 사용되는 성능 평가 기준인 F-measure를 적용한다. F-measure는 정확률과 재현율을 함께 고려하여 성능을 측정하는 도구로 다음과 같이 평가 값을 얻는다. 본 논문에서는 정확률과 재현율을 동등하게 고려하기위해 F-measure의 α값을 1로 설정하였다.

$$F-measure=\frac{(a+1)\times precision\times recall}{a\times precision+recall}$$

$$precision=\frac{A}{A+B},\quad recall=\frac{A}{A+C}$$

입력질의 / 패턴-답변쌍	포함	미포함
포함	A	B
미포함	C	D

A, B, C, D : 빈도 수

하지만 비슷한 질의문이 다른 스크립트에도 존재할 수 있으므로, 순차패턴매칭만으로는 적절한 스크립트를 선택할 수 없다. 본 논문에서는 적절한 스크립트를 선택하기 위해서 주제 큐를 유지한다. 주제 큐는 대화의 히스토리를 저장하는 역할을 하며, 스크립트 선택시 pre_topic 태그의 정보와 주제 큐의 정보를 비교하여, 일치하는 주제가 존재하면 해당 스크립트에 가중치를 부여한다. pre_topic에 의해 가중치가 부여된 최종 스크립트 평가점수는 다음과 같다.

$$Score_s=F_s+F_s\times w^t$$

F_s는 순차패턴매칭 점수를 나타내며, w는 부여할 가중치 정도, t는 일치하는 주제의 주제 큐에서의 최근 입력과의 거리를 나타낸다.

3.2 대화 흐름 제어

대화를 진행하다 보면 하나의 대화가 완료되기 전에 다른 대화라 발화되기도 하고, 상황에 따라 기억해야할 변수의 값이 달라진다. 본 논문에서는 대화 에이전트가 이러한 상황을 모델링할 수 있도록 주제 스택과 그에 따른 템플릿 변수 스택을 유지한다.

주제 스택은 현재 진행중인 대화 외에 다른 대화가 입력될 경우, 혹은 sub_topic 태그를 만났을 경우 새로운 주제가 푸쉬(push)되며, end_topic 태그를 만날 경우, 팝(pop)된다. 이러한 주제 스택을 이용하면, 서브 대화가 종료되었을 때, 다음으로 실행해야할 대화를 쉽게 찾아낼 수 있어 원활한 대화 흐름을 유도할 수 있다.

템플릿 변수의 경우는 한 주제에 대한 대화가 끝남에 따라서 그 정보가 유지되어야 하는 경우도 있고, 그렇지

않은 경우도 있다. 본 논문에서는 정보의 중복(한 변수의 값이 두 개를 갖게 되는 경우)이 발생하지 않는 한, 이전 값을 유지하도록 하며, 중복된 정보에 대해서는 사용자 정의 함수를 통해 정보를 관리하도록 한다. 이를 위해, 템플릿 변수 스택에는 중복되는 주제 스택이 들어올 때마다 현재 템플릿 변수의 값을 저장하고, 중복된 주제가 주제 스택에서 팝되면, 템플릿 변수를 저장했던 시점의 값으로 롤백(roll-back)한다.

3.3 상호주도형 대화

제안하는 스크립트 언어는 기본적으로는 사용자주도형 대화를 기반으로 동작하며, 다음의 세 경우에 대해서는 시스템주도형 대화가 발현된다. 첫째는 사용자의 질의문이 모호하여 하나 이상의 스크립트가 선택된 경우로 에이전트는 스크립트에 명시된 topic_question을 수행함으로써 모호함을 해결할 질의문을 얻는다. 둘째는 스크립트의 행동을 수행하기에는 정보가 부족한 경우로, 에이전트는 necessary_var 태그에서 정보가 필요한 템플릿 변수를 선택해 해당 템플릿 변수의 tpl_question을 수행함으로써 질의문을 얻는다. 마지막은 에이전트가 행동을 수행하는 중에 extract_candidate_var 태그를 만나는 경우로, 두 번째 경우와 비슷하게 에이전트는 candidate_var 태그에서 템플릿 변수를 선택해 질의문을 얻는다.

4. 실험 및 결과

제안하는 방법의 가능성을 평가하기 위해, 제안한 스크립트 언어를 이용하여 일정관리 도메인에 적용해 보았다.

표 4. 상호주도형 대화 예

1) 사용자: 오늘 식사 약속이 생겼어.
스크립트 선택: topic-일정입력
정보 추출: 일정종류-식사, 일정날짜-오늘
에이전트: 누구랑 만나기로 하셨나요?
정보 획득(질의): 일정대상
2) 사용자: 가족들이랑.
스크립트 선택: topic-일정입력
정보 추출: 일정대상-가족
에이전트: 신촌에서 보시나요?
정보 획득(추천): 일정장소
sub_topic: 일정추천
3) 사용자: 어
스크립트 선택: topic-일정장소추천
정보 추출: 일정장소-신촌
에이전트: 몇 시 약속인가요?
정보 획득(질의): 일정시간
4) 사용자: 1시에 보기로 했어.
스크립트 선택: topic-일정입력
정보 추출: 일정시간-1시
에이전트: 일정을 등록하였습니다.
서비스 제공: 일정등록

표 5. 템플릿 변수 표현 예

```
<template>
  <object> 일정 </object>
  <variable> $장소$ </variable>
  <value> 신촌 </value>
  <value> 서울역 </value>
  ...
  <tpl_question>
    <function>    recommend_location|$일정$날짜$|$일정
$시간$|$일정$대상$|$일정$종류$
      <return value = false>
        <answer> 어디에서 모이나요? </answer>
      </return>
      <return value = true>
        <function>  answer_recommend_location
                    |&$internal$ans$ </function>
        <answer> $internal$ans$ </answer>
        <sub_topic> 일정장소추천 </sub_topic>
      </return>
    </function>
  </tpl_question>
</template>
```

표 6. 스크립트 예

```
<script>
  <topic>일정입력</topic>
  <necessary_var>
    <variable> $일정$날짜$ </variable>
    <variable> $일정$시간$ </variable>
    <variable> $일정$유형$ </variable>
    <variable> $일정$대상$ </variable>
    <variable> $일정$장소$ </variable>
  </necessary_var>
  <query>$일정$날짜$      $일정$종류$가에      생겼어.
</query>
  <query>$일정$대상$이랑. </query>
  <query>$일정$시간$에 보기로 했어. </query>
  ...
  <action>
    <function> SchedInput|$일정$날짜$|$일정$시간$|$일
정$대상$|$일정$종류$|$일정$장소$
      <return value=true>
        <answer> 일정을 등록하였습니다. </answer>
        <end_script/>
      </return>
      <return value=conflict>
        <function>
          QuesSchedInputConflict|&$internal$answer$
        </function>
        <answer> $internal$answer$ </answer>
        <change_topic> 일정충돌 </change_topic>
      </return>
    </function>
  </action>
  <topic_question> 일정입력하시게요? </topic_question>
</script>
```

표 4는 상호주도형 대화의 예를 보여준다. 대화 1)에서 에이전트는 사용자의 입력문과 스크립트의 질의문간의 패턴매칭을 통해서 주제가 '일정입력'임을 알아내고, 필요한 정보를 추출한다. 그 후, 해당 스크립트에서 necessary_var이 모두 추출되었는지 확인하고, necessary_var이 모두 존재하지 않으므로 necessary_var 변수 중 하나를 선택하여 템플릿 변수의 tpl_question을 수행한다.

대화 2)에서는 대화 1)과 같은 동작을 반복하고 있으나, 특정 장소를 얻기 위해 표 5에서 정의된 추천함수를 구동시킨 결과를 얻어서 질의문을 만들었다. 표 5의 tpl_question을 보면 recommend_location 함수의 리턴값이 'false'이면 준비된 문장을 실행하도록 하며, 'true'이면 추천결과 문장을 사용자 지정 함수를 통해서 받아옴을 알 수 있다. 대화 2)에서는 추천함수 실행 결과가 'true'이고 이에 대한 답변문이 '신촌에서 보시나요?'임을 보여준다. 다시 표 5로 돌아와서 리턴값이 'true'인 경우 answer 태그가 수행된 후에 일정추천을 sub_topic에 넣어주는 것을 볼 수 있다. 이는 대화의 주제가 '일정장소추천'상태로 변경했음을 명시하여, 다음에 올 사용자의 답변을 적절히 처리하기 위함이다.

대화 3에서 사용자 입력은 '어'이다. 스크립트에는 시스템주도형 대화에 대한 사용자 답변으로 '어'에 해당하는 많은 스크립트가 존재하지만, 대화 2)에서 sub_topic으로 '일정장소추천'을 넣어주어 이 대화에서는 일정장소추천에 대한 답변으로 잘 처리되고 있다. 스크립트 예에서는 나타나 있지 않지만, '일정장소추천'에 대한 사용자 답변이 '어'인 경우 end_script가 수행된다. 주제 스택에는 대화 1)에서 '일정입력'이 들어왔고, 대화 2)에서 '일정장소추천'이 들어왔으므로, end_script를 만나게되면 주제 스택의 top에는 다시 '일정입력'이 위치하게 된다. 따라서 에이전트는 '일정입력'에 대한 스크립트로 돌아가 계속적으로 necessary_var의 정보를 수집한다.

마지막으로 대화 4에서는 일정등록에 필요한 모든 변수가 추출되어 실제 서비스가 동작함을 보여준다. 표 5는 표 4의 대화에서 사용된 템플릿 변수를, 표 6은 표 4의 대화를 위한 스크립트의 일부를 보여준다.

5. 결론 및 향후 연구

본 논문에서는 상호주도형 대화 에이전트 구현을 위한 도메인 독립적 스크립트 언어를 제안하였다. 제안한 방법에서는 스크립트의 necessary_var 태그와 candidate_var 태그, topic_question 태그, 그리고 템플릿 변수의tpl_question 태그를 통해서 사용자주도형 대화뿐만 아니라 시스템주도형 대화도 발생할 수 있게 하였고, 도메인과 관련된 템플릿 변수의 정의 및 함수의 정의를 통해서 특정 도메인에서 동작할 수 있는 에이전트를 설계 할 수 있도록 하였다.

향후 연구로는 다양한 도메인에 제안한 스크립트 언어를 적용해 대화 에이전트를 구현하여 제안한 방법의 유용성에 대한 검증이 필요하며, 보다 유연한 대화의 진행을 위해 확률적 모델을 이용한 주제의 흐름 제어기법이 필요하다.

감사의 글

이 연구는 산업자원부가 지원한 뇌과학 연구 프로그램에 의해 지원되었음

참고문헌

[1] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu and A. Stent, "Towards conversational human-computer interaction," *AI Magazine*, vol. 22, no. 4, pp. 27-38, 2001.

[2] P. Nugues, C. Godereaux, P.O. El-Guedj, and F. Revolta, "A conversational agent to navigate in virtual worlds," *Proc. 11th TWLT, Dialogue Management in Natural Language Systems*, pp. 23-33, 1996.

[3] H. M. Meng, C. Wai and R. Pieraccini, "The use of belief networks for mixed-initiative dialog modeling," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, pp. 757-773, 2003.

[4] E. Levin, R. Pieraccini and W. Eckert, "A stochastic model of human-machine interaction for learning dialogue strategies," *IEEE Transaction on Speech Audio Processing*, vol. 8, pp. 11-23, 2000.

[5] M. Walker, "An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email," *Journal of Artificial Intelligence Research*, vol. 12, pp. 387-416, 2000.

[6] K. Kita, Y. Fukui, M. Nagata and T. Morimoto, "Automatic acquisition of probabilistic dialogue models," *Proce. of Fourth Int. Conf. of Spoken Language Processing*, vol. 1, pp. 196-199, 1996.

[7] J.-H. Hong, Y.-S. Song, and S.-B. Cho, "A hierarchical Bayesian network for mixed-initiative human-robot interaction," *Proc. of IEEE, Int. Conf. on Robotics and Automation*, pp. 3808-3813, 2005.

[8] E. Horvitz, J. Breese, D. Heckerman, D. Hovel and K. Rommelse, "The lumiere project: Bayesian user modeling for inferring the goals and needs of software users," *Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence*, pp. 256-265, 1998.

[9] G. Ferguson, J. Allen and B. Miller, "TRAINS- 95: Towards a mixed-initiative planning assistant," *Proc. of the Third Conf. on Artificial Intelligence Planning Systems*, pp. 70-77, 1996.