대화형 유전 프로그래밍을 이용한 적응적 문장생성 열차예약 에이전트

(Train Booking Agent with Adaptive Sentence Generation Using Interactive Genetic Programming)

임 성 수 † 조 성 배 ††
(Sungsoo Lim) (Sung-Bae Cho)

요 약 대화형 에이전트가 다양한 분야에서 적용됨에 따라서 현실성 있는 대화 생성을 위한 자연언어 생성에 대한 연구가 관심을 끌고 있다. 대화형 에이전트에서는 보통 미리 준비된 답변을 이용하여 사용자와 대화를 수행하지만, 최근에는 문장을 동적으로 생성하고 학습함으로써 보다 유연하고 현실성있는 서비스를 제공하는 대화형 에이전트가 활발히 연구되고 있다. 본 논문에서는 대화형 유전 프로그래밍을 이용한 문장생성 방법을 제안한다. 이 방법은 문장의 구조를 나타내는 문장계획 트리로 인코딩된 개체를 평가자의 평가를 통해 적응적인 문장을 얻는다. 이 방법의 유용성을 검증하기 위해 제안하는 방법으로 열차예약 에이전트를 구현한 후, 사용자 평가를 수행하였다. 그 결과 제안하는 방법이 도메인에 적합한 문장을 생성하는 것을 확인할 수 있었다.

키워드 : 대화형 에이전트, 자연언어 생성, 대화형 유전 프로그래밍, 문장계획 트리

Abstract As dialogue systems are widely required, the research on natural language generation in dialogue has raised attention. Contrary to conventional dialogue systems that reply to the user with a set of predefined answers, a newly developed dialogue system generates them dynamically and trains the answers to support more flexible and customized dialogues with humans. This paper proposes an evolutionary method for generating sentences using interactive genetic programming. Sentence plan trees, which stand for the sentence structures, are adopted as the representation of genetic programming. With interactive evolution process with the user, a set of customized sentence structures is obtained. The proposed method applies to a dialogue-based train booking agent and the usability test demonstrates the usefulness of the proposed method.

Key words : Conversational agent, Natural language generation, Interactive genetic programming, Sentence plan tree

1. 서 론

대화형 에이전트는 인간이 사용하는 자연언어로 해당 분야의 전문 지식을 알려주는 메신저 기반의 대화형 전문가 시스템이다[1]. 전통적인 대화 시스템은 다양한 문장생성보다 사용자 입력 문장의 이해나 대화 흐름의 제어에 초점을 두었으나[2], 최근 몇 년간 대화형 에이전트가 다양한 분야에서 사용이 증가함에 따라서 자연언어 생성(natural language generation)에 대한 관심도 증가하고 있다. 기존의 대화형 에이전트는 고정된 스크립트나 템플릿을 기반으로 사용자와 대화를 진행하기 때문에 유연하고 실용적인 서비스를 제공하지 못했다[3]. 이러한 문제점을 극복하기 위해 최근 연구에서는 대화 시스템에 자연언어 생성 기능을 부가시키고 있다.

자연언어 생성은 비언어적 정보표현으로부터 문장을 생성하는 방법으로, 전문가 시스템에서 사용자에게 정보를 제공하기 위한 수단으로 사용되기도 하였다[4]. 이러한 자연언어 생성은 문장생성 과정 중 적절한 단어의 선택과 어순의 선택 등 많은 선택의 문제들이 있기 때문에 어렵다.

전통적인 자연언어 생성에서는 문법이나 템플릿에 기

· 이 연구는 산업자원부가 지원하는 뇌과학 연구 프로그램에 의해 지원되었음

† 학생회원 : 연세대학교 컴퓨터과학과
lss@sclab.yonsei.ac.kr
†† 정 회 원 : 연세대학교 컴퓨터과학과 교수
sbcho@cs.yonsei.ac.kr

논문접수 : 2005년 4월 14일
심사완료 : 2005년 12월 13일

반한 문장생성 방식을 사용하였다. 특히 템플릿에 기반한 방식의 경우는 구현하기가 쉽고 특정 도메인에서 높은 성능을 보이기 때문에 많은 대화 시스템에서 적용되었다. 하지만 시스템의 설계자는 문법기반 방식의 경우 잘못된 문장이 생성되지 않도록 주의 깊게 문법을 디자인해야 하고, 템플릿기반의 경우 다양한 종류의 문장들을 지원할 수 있도록 많은 양의 대화 템플릿을 구축해야 한다. 그러나 설계자가 모든 가능한 문장구조를 지원하는 문법이나 템플릿을 준비한다는 것은 불가능하다[5]. 이와 같은 한계점을 극복하기 위해서 최근에 학습을 통한 문장생성 방법이 시도되고 있다[3,5-7]. Walker 등은 문장의 구조를 표현하는 문장계획 트리(Sentence plan tree; SPT)를 사용한 방법을 제안하고 랭크부스트(RankBoost) 방법을 적용하여 학습하였으며[3], Levin 등은 문장생성 기법을 위하여 통계적 방법을 제안하고 이를 대화기반 여행계획 시스템에 적용하였다[6]. 그리고 Ratnaparkhi는 결합모델(hybrid model)을 제안하였고[5], Bulyko와 Ostendorf는 가중 유한상태 기계(weighted finite state machine)를 제안하였다[7].

본 논문에서는 대화 시스템에서 적응적 문장을 생성하기 위한 대화형 유전 프로그래밍 기법을 제안한다. 대화형 유전 프로그래밍은 대화형 유전 알고리즘과 같이 평가자로부터 유전자 트리의 적합도 값을 평가 받아 트리 구조를 진화시키는 방법으로 본 논문에서는 문장계획 트리를 사용하여 유전자 트리를 표현하였다.

본 논문의 구성은 다음과 같다. 2절에서는 기존의 자연언어 생성에 관련한 연구들을 소개하고 3절에서는 문장계획 트리와 유전 프로그래밍이 어떻게 문장을 만들어 가는지를 설명하며, 4절에서는 제안하는 방법으로 열차예약 시스템을 구현한다. 5절에서는 이 시스템의 유용성을 검증하기 위한 실험 내용과 결과를 논의하고, 마지막으로 6절에서는 결론 및 향후 연구에 대해 언급한다.

2. 자연언어 생성

자연언어 생성은 컴퓨터 프로그램이 정보의 내부 표현 방식으로부터 어떻게 자연언어 텍스트를 생성할 수 있는가를 연구하는 것이다[8,9]. 최근 사람과 컴퓨터간 대화 시스템의 가능성이 증가함에 따라서 다양한 문장생성을 통한 자연스러운 답변 제공의 방법이 주목받고 있다. 자연언어 생성과 관련한 많은 연구에서는 구문론 문법(syntactic grammar)을 이용한 파서와 같은 생성문법 규칙을 사용한다[10]. SURGE는 이러한 규칙기반 시스템의 좋은 예이다[11]. SURGE를 사용한 많은 응용 어플리케이션은 일반적으로 잘 작성된 생성문법 규칙이 자연언어 생성시스템의 다양한 도메인에서 적용 가능성과 재사용성, 도메인 독립성을 제공함을 입증하였다[10].

그러나 SURGE 및 기타 다른 규칙기반 시스템은 각 단어의 속성, 분류, 정의 등을 설계자가 디자인해야 하므로 많은 노력이 필요하다는 단점이 있다. 규칙기반 시스템이 템플릿 시스템에 비해서 더 좋은 성능을 보이지만 생성문법 정의에 있어서 많은 시간과 노력이 필요하기 때문에 대부분의 대화 시스템에서는 템플릿을 사용한다.

템플릿기반 방식은 문장의 일부분을 변수로 놓고 상황에 따라 변수 값을 적절한 값으로 대체하여 문장을 생성하는 방법이다. 이 방법은 미리 구축된 템플릿을 기반으로 문장이 생성되므로 얼마나 많은 양의 템플릿을 구축했는지가 시스템의 성능을 좌우한다. 현재 많은 연구에서 템플릿기반 방식을 사용하는데 이것은 템플릿기반 방식이 짧은 시간에 높은 성능을 내는 시스템을 구축하기 쉽기 때문이다. 그러나 템플릿기반 방식은 특정 상황을 가정하여 구성하기 때문에 재사용성이 떨어지며, 동일하거나 비슷한 패턴의 대화가 계속되어 유연성이 부족하다[12].

최근 자연언어 생성에 학습기법을 도입하여 규칙기반 시스템과 템플릿기반 시스템의 한계를 극복하려고 시도되었다. Ratnaparkhi는 자연언어 생성의 네 가지 학습기법을 제안하고 여러 기법들을 비교 평가하였다[5]. NLG1은 기준선을 이용한 것으로 간단하게 훈련 데이타 집합에서 가장 높은 빈도로 출현한 템플릿을 선택하는 것이고, NLG2는 주어진 속성-값(attribute-value) 집합에서 가장 좋은 표현을 할 수 있는 선택은 입력 속성에서 높은 확률을 갖는 연속된 단어의 순서라고 가정한다. 그리고 NLG3은 단어 히스토리에서 가까운 두 개의 단어만을 참고하는 NLG2의 결점에 초점을 두고 NLG2의 좌우 단어의 순서를 고려한 방식과는 달리 구문구조 트리에서 단어들의 상하 의존관계를 고려하여 문장을 생성한다. 그리고 NLG4는 문법과 통계적인 방법을 결합하여 문장을 생성한다.

그밖에 Bangalore와 Rambow는 의존문법(dependency grammar)을 통계적인 정보와 함께 사용하였고[13], Walker 등은 항공여행 도메인에서 문장계획 트리를 이용한 자연언어 생성 방법을 제안하였다[3]. 그리고 Oh와 Rudnicky는 항공여행 도메인에서의 통계적인 접근방법을 제안하였다[10].

본 논문에서는 대화형 유전 프로그래밍을 이용하여 문장을 생성하는 방법을 제안한다. 이전 연구에서는 문장의 문법구조를 BNF(Backus Naur Form)로 나타내고 문법의 파스 트리를 유전자 트리로 표현하였다[14]. 이 방법은 진화를 통해서 기존의 규칙기반 시스템에서 문법 설계의 어려움을 극복하고 사용자 적응적 문장을

생성해 내고자 시도되었다. 그러나 간단한 문법으로부터 불완전한 문장의 생성을 방지하려면 장시간의 진화가 필요하고 진화에 걸리는 시간을 줄이려면 잘 설계된 문법이 필요하다. 본 논문에서는 유전자 트리의 인코딩 방법으로 BNF대신 템플릿기반의 문장계획 트리를 사용하여 불완전한 문장이 생성되지 않으면서 보다 간결하게 시스템을 디자인할 수 있는 방법을 제안한다.

3. 유전 프로그래밍을 이용한 적응적 문장생성

유전 프로그래밍은 컴퓨터가 문제를 풀 수 있는 프로그램을 자동으로 생성하는 기법으로 John R. Koza가 제안했다. 이 방법은 유전 알고리즘의 변형으로 집단의 각 개체는 컴퓨터 프로그램을 나타낸다[15]. 본 논문에서는 문장생성 방법으로 문장계획 트리를 유전자 개체로 하는 대화형 유전 프로그래밍을 사용한다. 대화형 유전 프로그래밍은 평가자의 평가를 바탕으로 진화가 이루어지므로 이를 이용하면 동적이고 적응적인 문장을 생성할 수 있다. 그림 1은 대화형 유전 프로그래밍을 이용하여 문장이 생성되는 과정을 보여주고 있다.

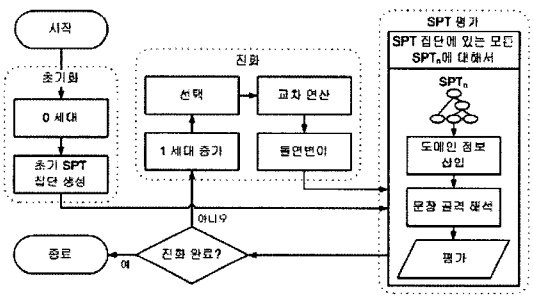

그림 1 대화형 유전 프로그래밍을 이용한 적응적 문장생성 과정

제안하는 방법에서의 유전자 개체는 문장계획 트리(Sentence plan tree, SPT)로 표현된다. SPT의 말단 노드는 하나의 도메인 정보를 갖는 템플릿으로 구성되고, 부모 노드는 하위 정보를 결합하기 위한 문장결합 연산자로 구성된다. SPT의 평가는 평가자에 의해서 직접 평가되며 평가자는 SPT 해석 과정을 통해서 해석된 문장을 보고 점수를 부여하게 된다.

3.1 문장계획 트리

본 논문에서는 유전자 트리의 표현 방법으로 문장계획 트리를 적용하였다. 문장계획 트리는 말단 노드에 존재하는 단문들을 부모 노드의 문장결합 연산자로 연결하여 복문을 만드는 방법으로 Lavoie와 Rambow가 고안하였고[16] Danlos, Gardent와 Webber 및 Stone과 Doran 등에 의해서 유사하게 사용되었다[17-19]. 이러한 문장계획 트리 T의 노드 N_k는 말단 노드의 경우에 미리 정의된 템플릿으로 구성되고, 말단 노드가 아닌 경우에는 두 개의 자식 노드 C_k^1, C_k^2를 가지며 두 서브트리 $T_{sub_k}^1$, $T_{sub_k}^2$가 나타내는 문장 S_k^1, S_k^2를 연결하는 결합 연산자 JO_i로 구성된다. 여기서 템플릿은 '당신은 $도착지$에 갑니다'와 같이 하나의 주어와 하나의 도메인 정보 및 하나의 동사로 구성되며, 도메인 정보는 '$' 안에 있는 변수로 위치, 시간 등과 같이 템플릿이 적용된 대상 도메인으로부터 얻을 수 있는 정보를 나타낸다. 템플릿은 제안한 방법을 적용한 시스템이 템플릿의 도메인 정보를 가지고 있는 경우는 평서형의 문장이 되며, 그렇지 않은 경우에는 해당 도메인 정보를 묻는 질문형의 문장이 된다. 문장결합 연산자로는 Walker 등의 연구를 바탕으로 한국어를 분석하여 $JO_{1\sim5}$의 총 5개를 정의하였다[3]. 각 결합 연산자는 평문과 평문의 결합, 평문과 질문의 결합, 질문과 질문의 결합에 적용되며 노드 N_k가 JO_i일 때, 다음과 같은 역할을 한다.

- JO_1 : 가장 간단한 방법으로 두 문장을 결합하는 연산자로 두 문장 S_k^1, S_k^2를 접속사 '그리고'로 연결한다.
- JO_2 : 두 문장 S_k^1, S_k^2가 동일한 주어를 가질 경우에 두 문장을 연결하는 연산자이다. S_k^2의 주어를 생략하고 접속사 '그리고'를 사용하여 두 문장을 결합한다.
- JO_3 : 두 문장 S_{k1}, S_{k2}가 동일한 주어와 동일한 동사를 가질 경우에 두 문장을 연결하는 연산자이다. S_{k2}의 주어와 동사는 생략되며, 접속사 '그리고'가 사용된다.
- JO_4 : 두 문장 S_{k1}, S_{k2}의 주어는 다르고 나머지 부분은 같은 경우 사용되는 연산자로 S_{k1}과 S_{k2}의 주어를 하나로 합치고 나머지 부분을 넣어줌으로써 두 문장을 결합한다.
- JO_5 : 두 문장 S_{k1}, S_{k2}가 동일한 주어를 가지며 다른 동사를 가지지만 두 동사의 의미를 포함할 수 있는 하나의 동사가 존재할 경우, 이 동사를 사용하여 두 문장을 결합한다.

표 1~3은 문장결합 연산자의 적용 예를 보여주고 있다.

문장계획 트리는 그림 1과 같이 SPT 해석 과정을 통해 평가자에게 전달된다. 먼저 도메인 정보 삽입기에서 SPT의 말단 노드인 단문과 도메인 정보를 합쳐 하나의 문장을 생성하고 문장 골격 해석기를 통해서 하나의 복문이 완성된다. 그림 2는 문장계획 트리의 해석과정을 보여주고 있다.

표 1 평문과 평문의 결합

Joint Operator	Statement A	Statement B
	Result statement	
JO 1	당신은 $도착지1$에 갑니다.	그는 $도착지2$에 갑니다.
	당신은 $도착지1$에 가고 그는 $도착지2$에 갑니다.	
JO 2	당신은 $사물1$을 좋아합니다.	당신은 $사물2$를 싫어합니다.
	당신은 $사물1$을 좋아하고 $사물2$를 싫어합니다.	
JO 3	그는 $시간$에 떠납니다.	그는 $출발지$를 떠납니다.
	그는 $시간$에 $출발지$를 떠납니다.	
JO 4	그는 $도착지$에 갑니다.	그녀는 $도착지$에 갑니다.
	그들은 $도착지$에 갑니다.	
JO 5	당신은 $출발지$를 떠납니다.	당신은 $도착지$에 도착합니다.
	당신은 $출발지$에서 $도착지$로 갑니다.	

표 2 평문과 질문의 결합

Joint Operator	Statement A	Question B
	Result question	
JO 1	당신은 $도착지$에 갑니다.	그는 어디에 가나요?
	당신은 $도착지$에 가고 그는 어디에 가나요?	
JO 2	당신은 $사물$을 좋아합니다.	당신은 무엇을 싫어하나요?
	당신은 $사물$을 좋아하고 무엇을 싫어하나요?	
JO 3	그는 $시간$에 떠납니다.	그는 어디를 떠나나요?
	그는 $시간$에 어디를 떠나나요?	
JO 4	그는 $도착지$에 갑니다.	그녀는 어디에 가나요?
	불가능 (도메인 정보 불일치)	
JO 5	당신은 $출발지$를 떠납니다.	당신은 어디에 도착하나요?
	당신은 $출발지$에서 어디로 가나요?	

표 3 질문과 질문의 결합

Joint Operator	Question A	Question B
	Result question	
JO 1	당신은 어디에 가나요?	그는 어디에 가나요?
	당신은 어디에 가고 그는 어디에 가나요?	
JO 2	당신은 무엇을 좋아하나요?	당신은 무엇을 싫어하나요?
	당신은 무엇을 좋아하고 무엇을 싫어하나요?	
JO 3	그는 몇 시에 떠나나요?	그는 어디를 떠나나요?
	그는 몇 시에 어디를 떠나나요?	
JO 4	그는 어디에 가나요?	그녀는 어디에 가나요?
	그들은 어디에 가나요?	
JO 5	당신은 어디를 떠나나요?	당신은 어디에 도착하나요?
	당신은 어디에서 어디로 가나요?	

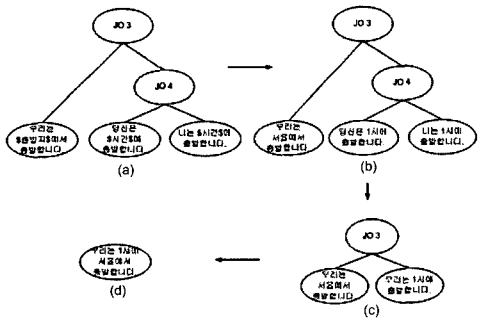

그림 2 SPT의 해석 과정

그림 2(a)의 말단 노드는 도메인 정보 삽입기에서 $출발지$로 '서울'과 $시간$으로 '1시'의 도메인 정보를 가지고 그림 2(b)와 같이 '우리는 서울에서 출발한다,' '당신은 1시에 출발한다,' '나는 1시에 출발한다'의 세 문장으로 변환된다. 그리고 문장 골격 생성기를 통해서 그림 2(b)의 두 문장 '당신은 1시에 출발한다'와 '나는 1시에 출발한다'는 결합 연산자 JO_4에 의해서 '우리는 1시에 출발한다'가 되며(그림 2(c)), 남은 두 문장은 결합 연산자 JO_3에 의해서 최종적으로 '우리는 1시에 서울에서 출발한다' 라는 문장이 생성된다(그림 2(d)).

3.2 문장계획 트리의 진화

그림 1에서 보듯이 문장계획 트리로 이루어진 유전자 트리들은 유전 프로그래밍의 교차연산자와 돌연변이 연산자를 통해서 다양한 개체들로 생성되며 선택 연산자를 통하여 평가자로부터 높은 점수를 얻은 개체들이 다음 세대에 살아 남게 된다. 유전자 트리의 생성 및 교차와 돌연변이 연산은 자연스러운 문장의 표현을 위해 다음과 같은 제약 조건을 갖는다.

- 말단 노드에 동일한 도메인 정보를 갖는 템플릿이 존재하지 않는다.
- 트리의 말단 노드의 수는 너무 길고 부자연스러운 문장이 생성되지 않도록 n개로 제한한다.
- JO_2는 두 문장 S_k^1, S_k^2가 동일한 주어를 가질 경우에만 사용된다.
- JO_3은 두 문장 S_k^1, S_k^2가 동일한 주어와 동일한 동사를 가질 경우에만 사용된다.
- JO_4는 두 문장 S_k^1, S_k^2의 주어는 다르고 나머지 부분은 같은 경우에만 사용된다.
- JO_5는 두 문장 S_k^1, S_k^2가 동일한 주어를 가지며 S_k^1, S_k^2의 두 동사를 합칠 수 있는 동사가 정의되었을 경우에만 사용된다.

그림 3과 그림 4는 문장계획 트리의 교차와 돌연변이 연산의 예를 보여준다. 그림 3의 (a)와 (b)는 SPT의 교차연산 전의 두 트리를 보여주고 있으며 음영 노드는 교차연산을 통해서 변경될 노드들을 나타낸다. 그리고 (c)와 (d)는 두 트리의 교차연산 후의 상태를 보여주고 있다. 교차연산의 결과 두 문장 '당신은 $도착지$에 가고 당신은 $출발지$를 $시간$에 떠납니다'와 '당신은 $시간1$에 떠나고 $시간2$에 $등급$을 타고 도착합니다'는 '당신은 $도착지$에 가고 당신은 $시간1$에 떠나

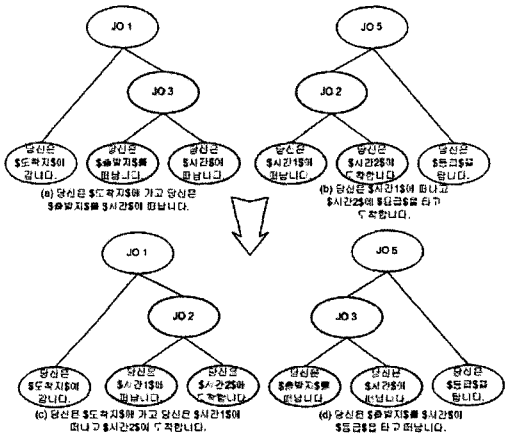

그림 3 교차연산자

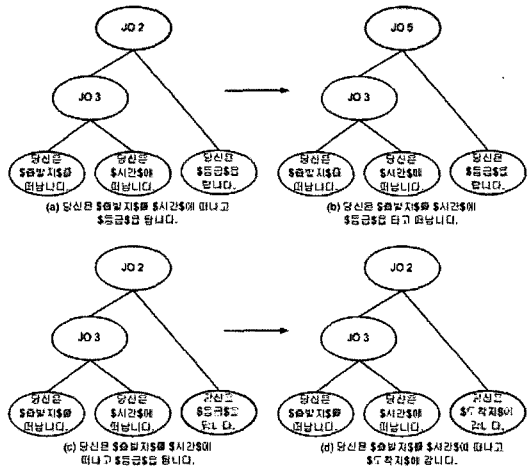

그림 4 돌연변이 연산자

고 $시간2$에 도착합니다'와 '당신은 $출발지$를 $시간$에 $등급$을 타고 떠납니다'의 두 문장으로 바뀐다.

그림 4의 (a)와 (c)는 돌연변이 전의 모습이고, (b)와 (d)는 돌연변이 후의 모습이다. 이 예는 돌연변이 연산자를 통해서 문장 '당신은 $출발지$를 $시간$에 떠나고 $등급$을 탑니다'가 '당신은 $출발지$를 $시간$에 $등급$을 타고 떠납니다'로 변경되고, 말단 단문의 돌연변이를 통해서 '당신은 $출발지$를 $시간$에 떠나고 $도착지$에 갑니다'로 변경되는 것을 보여준다.

3.3 문장의 다양성

여기에서는 문장계획 트리를 이용한 유전 프로그래밍에서 얼마나 다양한 문장을 생성하는지를 알아본다.

정리. 총 n_{JO}개의 문장결합 연산자가 존재할 때, 도메인 정보의 개수를 n_D라 하고 각 도메인 정보에 n_T개의 템플릿이 준비되어 있으며, 말단 노드의 개수를 n_D로 제한하고 문장결합 연산자가 어떠한 두 문장도 연결할 수 있다고 가정하면 제안하는 방법을 통해서 $\sum_{k=1}^{n_D} {}_{n_D}P_k \times {n_T}^k \times \frac{(2k-2)!}{k!(k-1)!}$개의 서로 다른 문장이 생성될 수 있다.

증명. 문장계획 트리에 k개의 말단 노드가 있다면 이러한 말단 노드를 구성할 수 있는 도메인 정보의 경우의 수는 ${}_{n_D}P_k$이고 말단 노드의 각 도메인 정보에 n_T개의 템플릿이 준비되어 있으므로 ${}_{n_D}P_k \times {n_T}^k$의 템플릿의 조합이 존재한다. 그리고 k개의 말단 노드를 가진 문장계획 트리는 k-1개의 문장결합 연산자가 존재하고 문장결합 연산자는 모든 문장을 연결할 수 있으므로 총 ${n_{JO}}^{k-1}$개의 문장결합 연산자 조합이 존재하고, 노드의

개수가 n개일 때 만들 수 있는 트리의 개수가 $\frac{(2n)!}{n!(n+1)!}$ 이 된다는 Catalan Number를 이용하면 말단 노드를 고려하지 않았을 경우, $n_{JO}^{k-1} \times \frac{(2k-2)!}{k!(k-1)!}$ 개의 서로 다른 트리 모양이 존재하게 된다. 여기에 말단 노드를 포함하면 k개의 말단 노드를 갖는 문장계획 트리에서는 총 ${}_{n_D}P_k \times {n_T}^k \times {n_{JO}}^{k-1} \times \frac{(2k-2)!}{k!(k-1)!}$ 개의 문장이 생성될 수 있다. 따라서 제안하는 방법이 생성할 수 있는 서로 다른 문장의 총 수는 $\sum_{k=1}^{n_D} {}_{n_D}P_k \times {n_T}^k \times {n_{JO}}^{k-1} \times \frac{(2k-2)!}{k!(k-1)!}$ 이 된다. □

표 4는 n_{JO}가 5일 때, 도메인 정보의 개수 n_D와 각 도메인 정보에 대한 템플릿의 개수 n_T에 따른 제안하는 방법이 생성할 수 있는 서로 다른 문장의 가지 수를 보여주고 있다.

4. 열차예약 에이전트

본 논문에서는 제안하는 방법의 유용성을 평가하기 위해서 열차예약 에이전트에 적용시켜보았다. 시스템의 개발 환경은 윈도우 XP에서 Visual C++ 6.0을 이용하였다. 열차예약 에이전트는 사용자와의 대화를 통하여 열차예약에 필요한 정보를 수집하고, 수집된 정보에 적합한 기차를 예약해 주는 시스템이다. 열차예약에 필요한 정보는 출발지, 도착지, 출발날짜, 출발시간, 도착시간 그리고 열차등급의 6가지로 구성된다. 기존의 고정된 스크립트를 사용하던 시스템에서는 준비된 문장을 바탕으로 사용자와 대화를 진행하므로 대화가 단순하며 기계적인 반면에 제안하는 시스템에서는 동적으로 다양한 문장을 생성하여 사용자와 대화를 진행함으로써 사용자에게 보다 친숙한 인터페이스를 제공한다. 그림 5는 열차예약 에이전트의 시스템 구성도를 보여주고 있다.

SPT 선택기는 에이전트가 사용자와 대화를 통해서 얻어낸 도메인 정보의 양에 따라서 문장생성에 적절한 SPT 그룹을 결정하고 해당 그룹에서 임의로 하나의 SPT를 선택하고, 사용자 평가를 바탕으로 SPT 그룹을

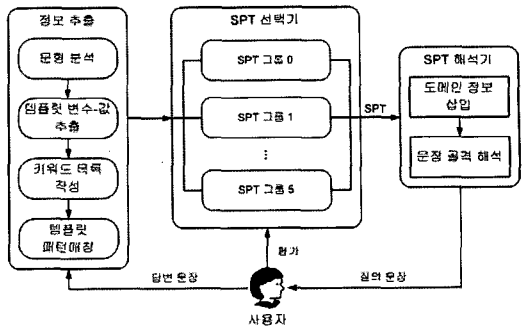

그림 5 열차예약 에이전트의 시스템 구성도

진화시킨다. SPT 그룹은 에이전트가 가지고 있는 도메인 정보의 양에 따라서 구분되며, 열차예약 에이전트에서는 총 6개의 문장 그룹이 존재하고 SPT의 진화는 각 그룹별로 이루어지는데, 이는 실제 대화에서 사용자에 대한 정보를 얼마나 아는가에 따라서 대화에 선택되는 문장이 달라지기 때문이다. 사용자의 출발시간을 묻는 질문의 경우 에이전트가 사용자의 정보를 아무것도 모른다면 "몇 시에 출발하시나요?" 정도의 질문만이 가능하지만, 에이전트가 사용자의 출발지가 '부산'인 것을 안다면 "부산에서 몇 시에 출발하시나요?"와 같은 문장을 사용하는 것이 에이전트가 사용자의 출발지를 정확하게 알고 있다는 것을 사용자에게 한 번 더 상기시킬 수 있고 에이전트가 잘못된 정보를 가지고 있을 경우 이를 바로 잡을 수 있는 기회를 얻을 수 있으므로 출발시간만을 묻는 것 보다는 더 좋은 문장이 된다. SPT 그룹 0은 에이전트가 도메인 정보를 하나도 알지 못할 경우에 사용되며, SPT 그룹 1은 도메인 정보가 한 개인 경우, SPT 그룹 2는 두 개인 경우 등과 같이 사용된다.

SPT 해석기는 SPT로부터 에이전트의 질의문을 이끌어 낸다. 표 5는 각 도메인 정보별로 SPT의 말단 노드에 들어갈 템플릿을 보여주고 있다. SPT에 있는 각각의 템플릿은 도메인 정보의 종류만을 표시하고, SPT 해석기에서 문장을 생성할 때 에이전트가 도메인 정보를 가지고 있을 경우는 평문 중에서 하나를 선택하여 문장을 만들고, 그렇지 않을 경우에는 질문 중에서 하나

표 4 도메인 정보와 템플릿의 개수에 따른 서로 다른 문장의 가지 수

n_D	n_T	문장 개수	n_D	n_T	문장 개수
3	2	2526	5	2	34,824,410
3	3	8,379	5	3	261,306,915
3	4	19,692	5	4	1,094,593,620
3	5	38,265	5	5	3,328,502,525
4	2	249,848	6	2	6,253,248,612
4	3	1,247,952	6	3	70,439,788,368
4	4	3,917,776	6	4	393,581,186,424
4	5	9,526,520	6	5	1,496,391,378,780

표 5 열차예약 에이전트에서 사용되는 단문

도메인 정보		단문
출발지	평문	손님은 $출발지$에서 출발합니다, 손님은 $출발지$를 떠납니다.
	질문	손님은 어디에서 출발하나요?
도착지	평문	손님은 $도착지$까지 가십니다, 손님은 $도착지$에 도착합니다.
	질문	손님은 어디까지 가시나요?
출발날짜	평문	손님은 $날짜$에 출발합니다, 손님은 $날짜$에 떠납니다, 손님은 $날짜$에 갑니다.
	질문	손님은 언제 출발하나요?, 손님은 언제 가나요?
출발시간	평문	손님은 $출발시간$에 출발합니다, 손님은 $출발시간$에 떠납니다.
	질문	손님은 몇 시에 출발하나요?, 손님은 몇 시에 가나요?
도착시간	평문	손님은 $도착시간$에 도착합니다.
	질문	손님은 몇 시에 도착하길 원하나요?
열차등급	평문	손님은 $열차등급$을 타십니다.
	질문	손님은 어떤 열차를 원하세요?

를 선택하여 문장을 만든다.

정보추출 단계에서는 SPT 해석기에서 생성된 질의에 대한 사용자의 답변을 패턴매칭과 템플릿을 사용하여 도메인 정보를 추출한다. 즉, 열차예약 에이전트는 사용자의 입력과 사용자 입력패턴을 인식하기 위해서 미리 정의된 템플릿들과의 패턴매칭을 통하여 가장 유사한 템플릿을 찾아내고, 이로부터 사용자의 도메인 정보를 얻는다. 예를 들어, 사용자가 '저는 서울에서 부산까지 여행합니다'라고 대답하였다면, 시스템은 형태소 분석을 통해서 '저,' '서울에서,' '부산까지,' '여행' 등의 키워드를 추출하고 도메인 정보 데이타베이스를 통해서 (1) '저,' '$출발지$에서,' '$출발지$까지,' '여행,' (2) '저,' '$출발지$에서,' '$도착지$까지,' '여행,' (3) '저,' '$도착지$에서,' '$출발지$까지,' '여행,' (4) '저,' '$도착지$에서,' '$도착지$까지,' '여행'의 총 4개의 후보 목록을 만든 후, 정의된 템플릿들 중에서 순차 패턴매칭을 통해 '저는 $출발지$에서 $도착지$까지 여행합니다'가 선택된다. 이로부터 시스템은 사용자의 출발지가 서울이며, 도착지가 부산인 것을 알아 낸다.

5. 실험 및 결과

본 논문에서는 제안하는 방법의 유용성을 보이기 위해 SPT 그룹의 진화 실험을 하였고 그 결과를 적용한 열차예약 에이전트와 사용자 간의 대화 예를 보여준다. 그리고 시스템 설계의 편의성, 문장의 다양성, 문장의 자연스러움의 관점에서 템플릿기반 시스템과 비교 평가하였다.

5.1 진화 실험

본 실험은 제안하는 방법이 생성한 문장이 진화 과정을 통해서 얼마나 피험자에게 만족감을 주는지 알아보기 위한 것으로 10명의 피험자들이 90세대에 걸쳐 6개의 SPT 그룹을 평가하였다. 피험자는 각 세대마다 생성되는 모든 문장을 평가해야 하므로 개체 수는 20개로 다소 적게 설정하였다. 그리고 문장 구조의 급격한 변화를 방기하기 위해서 교차연산 비율은 다소 낮은 0.6으로 설정하였고, 다양한 표현을 생성할 수 있도록 돌연변이 연산 비율은 다소 높은 0.2로 설정하였다.

그림 6은 진화가 진행됨에 따른 각 SPT 그룹별 평균 평가점수를 보여준다. 피험자는 각 세대에서 생성된 문장들을 0~10점으로 평가했다. 결과를 보면, 처음 10세대 동안은 점수가 증가하다가 이후 10세대는 감소하는 경향을 보인다. 이것은 처음에 생성된 어색한 문장이 진화를 통해서 초기에는 빠르게 진화하여 좋은 점수를 얻었으나 이 후에는 진화 속도가 피험자의 기대치보다 늦어 상대적으로 낮은 점수를 받았기 때문이다.

그림 7은 진화과정에서 생성된 문장계획 트리를 보여주고 표 6은 도메인 정보로 $출발지$, $도착지$, $출발날짜$를 각각 '서울,' '부산,' '10월 10일'을 사용했을 때 생성된 문장을 나타낸다. 표 6에서 보는 바와 같이 처음 세대에서는 복잡하고 균형 잡히지 않은 문장이 생성되었지만, 진화가 진행됨에 따라서 피험자가 보다 간결한

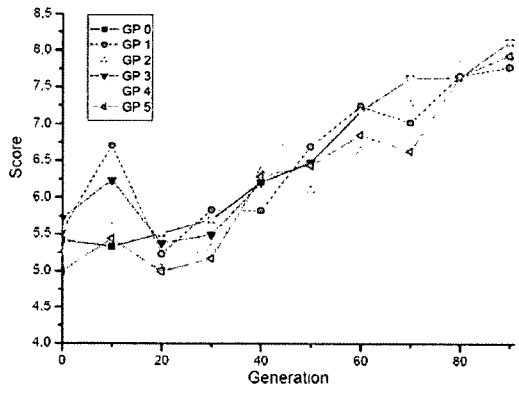

그림 6 세대별 진화 과정에 따른 평가 점수

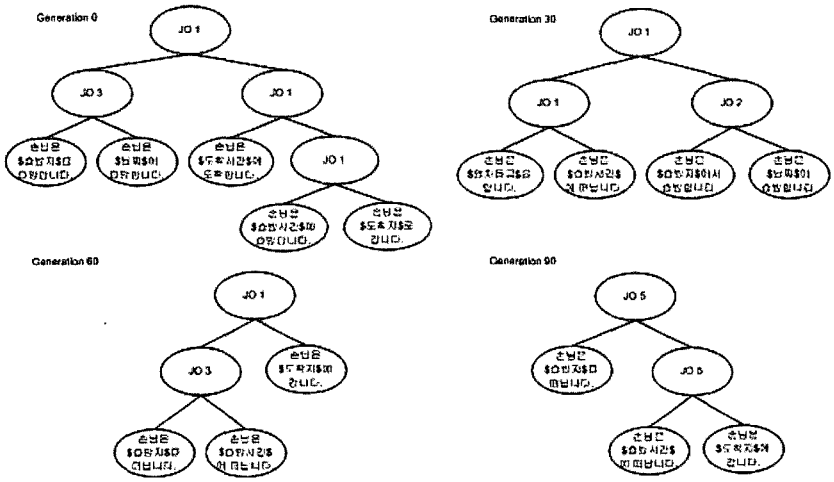

그림 7 열차예약 에이전트의 진화과정 중 생성된 문장계획 트리들

표 6 열차예약 에이전트의 진화과정 중 생성된 문장

세대	문장
0	손님은 서울에서 10월 10일에 출발하고 몇 시에 도착하길 원합니까? 그리고 몇 시에 도착하길 원하십니까? 그리고 손님은 부산에 갑니다.
30	손님은 서울에서 10월 10일에 출발합니다. 어떤 기차를 타길 원하십니까? 그리고 손님은 몇 시에 떠납니까?
60	손님은 부산에 갑니다. 그리고 손님은 서울을 몇 시에 떠나십니까?
90	손님은 서울에서 부산으로 몇 시에 갑니까?

표 7 대화 예

	문장	도메인 정보	비고
Agent	손님은 어디에서 어디로 가십니까?	없음	SPT 그룹 0 사용
User	서울에서 부산으로 갑니다.		정보 추출 $출발지$ 서울 $도착지$ 부산
Agent	언제 출발하나요?	$출발지$ 서울 $도착지$ 부산	SPT 그룹 2 사용
User	10월 10일요.		정보 추출 $출발날짜$ 10/10
Agent	서울에서 부산으로 몇 시에 출발하나요?	$출발지$ 서울 $도착지$ 부산 $출발날짜$ 10/10	SPT 그룹 3 사용
User	오후 10시쯤 출발할껀데요.		정보 추출 $출발시간$ 10:00 PM
Agent	어떤 종류의 기차를 원하시죠?	$출발지$ 서울 $도착지$ 부산 $출발날짜$ 10/10 $출발시간$ 10:00 PM	SPT 그룹 4 사용
User	KTX요		정보 추출 $열차등급$ KTX
Agent	10월 10일 서울에서 부산으로 가는 KTX 열차는 9:50 PM과 10:20PM에 있습니다.	$출발지$ 서울 $도착지$ 부산 $출발날짜$ 10/10 $출발시간$ 10:00 PM $열차등급$ KTX	검색 완료

문장에 높은 점수를 부여한 결과 간결한 문장을 얻을 수 있었다.

5.2 열차예약 에이전트와 사용자 간의 대화 예

표 7은 열차예약 에이전트와 사용자 간의 대화 예를

보여주고 있다.

5.3 템플릿기반 시스템과 비교 평가

본 실험은 문장생성 기법 중 가장 널리 사용되는 템플릿기반 방법과 제안하는 방법을 설계의 편의성과 문장의 다양성의 관점에서 비교하기 위해 총 12명의 피험자를 3개의 그룹으로 나누어 수행되었다. 첫 번째 그룹의 목적은 제안하는 방법을 적용한 시스템을 설계하는 것으로 템플릿 설계 및 진화 평가를 하였고, 두 번째 그룹의 목표는 템플릿기반 시스템을 설계하는 것으로 첫 번째 그룹의 템플릿 설계시간의 약 6배의 시간인 30분 동안 템플릿을 설계하였으며, 마지막 그룹은 첫 번째 그룹의 템플릿 설계에 걸린 시간과 진화 평가에 걸린 시간을 합친 시간과 비슷한 100분 동안 템플릿을 설계하였다. 그림 8은 첫 번째 그룹이 시스템을 설계하는데 걸린 시간을 보여주고, 그림 9는 피험자가 설계한 템플릿의 평균 개수를 보여주고 있다. 템플릿의 설계과정은 초기에는 단문 템플릿을 위주로 빠르게 템플릿을 추가할 수 있지만, 템플릿의 개수가 늘어감에 따라서 템플릿의 중복체크와 복합문장의 자연스러움 등을 고려해야 하므로 템플릿 추가의 속도가 현저하게 떨어졌다.

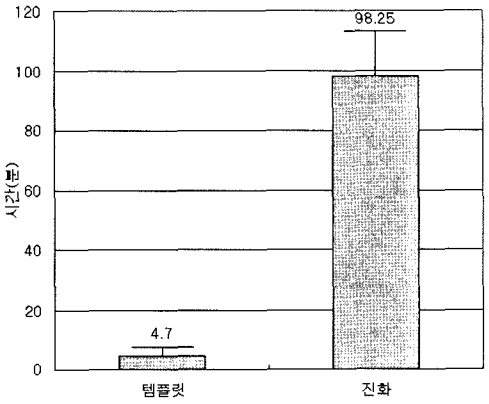

그림 8 제안하는 방법의 시스템 설계시 드는 시간 비용

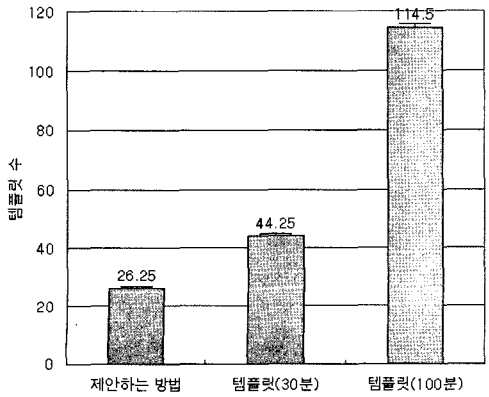

그림 9 설계된 템플릿의 개수

각 피험자는 시스템 설계가 끝난 후, 세 그룹의 시스템 중 자신이 설계하지 않은 시스템을 하나씩 사용해보고 시스템이 생성한 문장의 다양성에 대해 각각 1~5점으로 평가하였다. 그림 10은 제안하는 방법이 적은 템플릿을 가지고 다양한 문장을 생성한다는 3.3절의 내용을 지지하는 실험 결과이다.

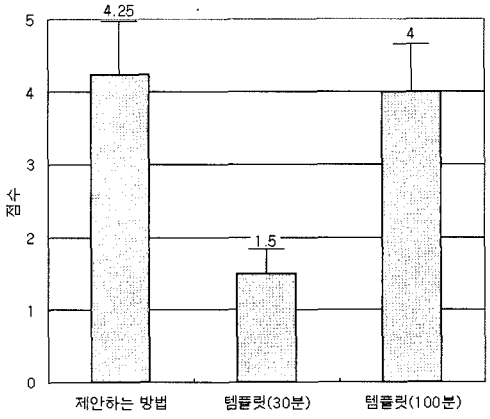

그림 10 문장의 다양성

6. 결 론

대화형 에이전트의 성능은 얼마나 다양한 대화를 수행할 수 있는지가 결정한다. 따라서 대화를 수행하기 위한 스크립트의 양이 많으면 많을수록 높은 성능의 대화 시스템을 구현할 수 있다. 그러나 많은 양의 스크립트를 준비하는 데에는 그만큼 많은 시간과 노력이 필요하다.

본 논문에서는 문장계획 트리를 대화형 유전 프로그래밍에 적용하여 사용자 적응적 문장을 생성하는 방법을 제안했다. 이 방법은 템플릿을 기반으로 하는 문장들을 사용자의 피드백을 받아서 학습하는 방법으로 이전 연구의 시스템 구축의 어려움을 해결하였고 문법을 대신하여 템플릿을 사용함으로써 잘못된 문법으로부터 불완전한 문장이 생성될 가능성을 없앴다. 그리고 학습 가능한 방법을 통해서 에이전트는 도메인 및 사용자에게 적응된 문장을 생성할 수 있었다.

참 고 문 헌

[1] S. Macskassy and S. Stevenson, "A conversational agent," *Master Essay*, Rutgers university, 1996.

[2] V. Zue and J. Class, "Conversational interfaces: Advances and challenges," *Proc. of the IEEE*, vol., 88, no. 8, pp. 1166-1180, 2000.

[3] M. A. Walker, O. C. Rambow and M. Rogati,

"Training a sentence planner for spoken dialogue using boosting," *Computer Speech and Language*, vol. 16, no. 3-4, pp. 409-433, 2002.

[4] M. Theune, "Natural language generation for dialogue: System survey," *TR-CTIT-03-22*, 2003.

[5] A. Ratnaparkhi, "Trainable approaches to surface natural language generation and their application to conversational dialog systems," *Computer Speech and Language*, vol. 16, no. 3-4, pp. 435-455, 2002.

[6] E. Levin, R. Pieraccini and W. Eckert., "A stochastic model of human-machine interaction for learning dialog strategies," *IEEE Trans. Speech and Audio Processing*, vol. 8, no. 1, pp. 11-23, 2000.

[7] I. Bulyko and M. Ostendorf, "Efficient integrated response generation from multiple targets using weighted finite state transducers," *Computer Speech and Language*, vol. 16, no. 3-4, pp. 533-550, 2002.

[8] W. Weiwei, L. Biqi, C. Fang and Y. Baozong, "A natural language generation system based on dynamic knowledge base," *Proc. of the 3rd Int. Conf. on ICSP*, pp. 765-768, 1996.

[9] K. McKeown, "Language generation: Applications, issues, and approaches," *Proc. of IEEE*, vol. 74, no. 7, pp. 905-919, 1986.

[10] H. Oh and I. Rudnicky, "Stochastic natural language generation for spoken dialog systems," *Computer Speech and Language*, vol. 16, no. 3-4, pp. 387-407, 2002.

[11] M. Elhadad and J. Robin, "An overview of surge: A reusable comprehensive syntactic realization component," *Technical Report 96-03*, Department of Mathematics and Computer Science, 1996.

[12] S. Seneff and J. Polifroni, "Formal and natural language generation in the Mercury conversational system," *Proc. of ICSLP*, vol. 2, pp. 767-770, 2000.

[13] S. Bangalore and O. Rambow, "Exploiting a probabilistic hierarchical model for generation," *Int. Conf. on COLING*, vol. 1, pp. 42-48, 2000.

[14] K.-M. Kim, S.-S, Lim and S.-B. Cho, "User adaptive answers generation for conversational agent using genetic programming," *IDEAL 2004, LNCS 3177*, pp. 813-819, 2004.

[15] J. Koza, *Genetic Programming: Automatic Discovery of Reusable Programs*, The MIT Press, 1994.

[16] B. Lavoie and O. Rambow, "A framework for customizable generation of multi-modal presentations," *COLING-AGL98*, pp. 718-722, 1998.

[17] L. Danlos, "G-TAG: A lexicalized formalism for text generation inspired by tree adjoining grammar," In Anne Abeille and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analysis, and Processing*, CSLI Publications, 2000.

[18] C. Gardent and B. Webber, "Varieties of ambiguity in incremental discourse processing," *Proc. of AMLap-98*, 1998.

[19] M. Stone and C. Doran, "Sentence planning as description using tree adjoining grammar," *ACE/EACL 97*, pp. 198-205, 1997.

임 성 수

2004년 2월 연세대학교 컴퓨터과학과 졸업(학사). 2006년 2월 연세대학교 컴퓨터과학과 졸업(석사). 2006년 3월~연세대학교 컴퓨터과학과 석사과정 재학중. 관심분야는 인공지능, 지능형 에이전트, 베이지안 네트워크

조 성 배

1988년 연세대학교 전산과학과(학사). 1990년 한국과학기술원 전산학과(석사). 1993년 한국과학기술원 전산학과(박사). 1993년~1995년 일본 ATR 인간정보통신연구소 객원 연구원. 1998년 호주 Univ. of New South Wales 초청연구원. 1995년~현재 연세대학교 컴퓨터과학과 정교수. 관심분야는 신경망, 패턴인식, 지능정보처리