문장계획 트리와 유전자 프로그래밍을 이용한 대화형 에이전트의 동적 문장생성

임성수[0], 조성배
연세대학교 컴퓨터과학과
lss@sclab.yonsei.ac.kr[0], sbcho@cs.yonsei.ac.kr

Dynamic Sentence Generation for a Conversational Agent Using Sentence Plan Tree and Genetic Programming

Sungsoo Lim[0] and Sung-Bae Cho
Dept. of Computer Science, Yonsei University

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 문장을 이용하여 사용자와 대화를 수행하지만, 최근에는 문장을 동적으로 생성하고 학습함으로써 보다 유연하고 현실성있는 서비스를 제공하는 대화형 에이전트가 활발히 연구되고 있다. 본 논문에서는 문장계획 트리를 인코딩 방법으로 적용한 대화형 유전자 프로그래밍을 통해 대화형 에이전트의 문장을 생성하는 방법을 제안한다. 피험자 12명을 대상으로 템플릿 기반 시스템과의 비교 실험결과, 제안하는 방법의 유용성을 확인할 수 있었다.

1. 서론

최근 몇 년간 대화형 에이전트가 여러 분야에 적용됨에 따라서 자연언어 생성에 대한 관심도 증가하고 있다. 전통적인 대화 시스템은 다양한 문장생성보다 사용자 입력 문장의 이해나 대화 흐름의 제어에 초점을 두었고[1], 고정된 스크립트나 템플릿을 기반으로 사용자와 대화를 진행하므로 유연하고 실용적인 서비스를 제공하지 못했다[2]. 이러한 문제점을 극복하기 위해 최근 연구에서는 대화 시스템에 자연언어 생성 기법을 적용시키고 있다.

본 논문에서는 대화형 유전자 프로그래밍 기법을 이용한 대화형 에이전트의 문장 생성 방법을 제안한다. 대화형 유전자 프로그래밍은 대화형 유전자 알고리즘과 같이 사용자로부터 유전자 트리의 적합도 값을 평가받아 트리 구조를 진화시키는 방법으로 본 논문에서는 문장계획 트리를 사용하여 유전자 트리를 표현하였다.

2. 자연언어 생성

자연언어 생성은 컴퓨터 프로그램이 정보의 내부 표현 방식으로부터 어떻게 자연언어 텍스트를 생성할 수 있는가를 연구하는 것이다[3]. 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[4]. SURGE는 이러한 규칙기반 시스템의 좋은 예로 SURGE를 사용한 많은 응용 어플리케이션은 일반적으로 잘 작성된 생성문법 규칙이 자연언어 생성시스템의 다양한 도메인에서 적용 가능성과 재사용성, 도메인 독립성을 제공함을 입증하였다[4]. 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[5].

최근 자연언어 생성에 학습기법을 도입하여 규칙기반 시스템과 템플릿기반 시스템의 한계를 극복하려고 시도되었다. Bangalore와 Rambow는 의존문법(dependency grammar)을 통계적인 정보와 함께 사용하였고[6], Walker 등은 항공여행 도메인에서 문장계획 트리를 이용한 자연언어 생성방법을 제안하였다[7].

3. 제안하는 방법

유전자 프로그래밍은 컴퓨터가 문제를 풀 수 있는 프로그램을 자동으로 생성하는 기법으로 John R. Koza가 제안했다. 이 방법은 유전자 알고리즘의 변형으로 집단의 각 개체는 컴퓨터 프로그램을 나타낸다[8]. 본 논문에서는 대화형 유전자 프로그래밍을 사용하여 동적이고 적응적인 문장을 생성한다. 그림 1은 대화형 유전자 프로그래밍을 이용하여 문장이 생성되는 과정을 보여주고 있다.

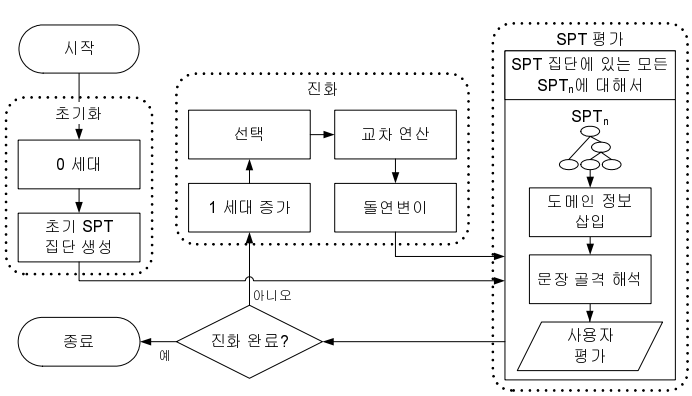

그림 1. 대화형 유전자 프로그래밍을 이용한 문장생성 과정

대화형 유전자 프로그래밍은 사용자가 직접 개체를 평가하도록 하는 방법이다. 제안하는 방법에서의 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 사용자에 의해서 직접 수행되며 사용자는 SPT 해석과정을 통해서 해석된 문장을 보고 점수를 부여한다.

3.1 문장계획 트리

문장계획 트리는 말단 노드에 존재하는 단문들을 부모 노드의 문장결합 연산자로 연결하여 복문을 만드는 방법으로 Lavoie와 Rambow가 고안하였다[9]. 본 논문에서는 문장계획 트리의 말단 노드를 템플릿으로 구성하여 주어진 상황에 맞춰 적절한 문장을 생성하도록 하였으며, 5개의 문장결합 연산자를 정의하였다.

템플릿은 '당신은 $도착지$에 갑니다.'와 같이 하나의 도메인 정보를 갖는 단문으로 구성되며, 여기서 도메인 정보는 '$'안에 있는 변수로 위치, 시간 등과 같이 템플릿이 적용된 대상 도메인으로부터 얻을 수 있는 정보를 나타낸다. 그리고 문장결합 연산자는 두 문장을 연결해주는 연산자로 본 논문에서는 다음과 같이 5개의 연산자를 정의하였다.

- JO_1: 가장 간단한 방법의 연산자로 두 문장을 접속사 '그리고'를 사용하여 연결한다.
- JO_2: 두 문장 A,B가 동일한 주어를 가질 경우, 문장 B의 주어를 생략하고 접속사 '그리고'를 사용하여 두 문장을 결합한다.
- JO_3: 두 문장 A, B가 동일한 주어와 동일한 동사를 가질 경우, 문장 B의 주어와 동사를 생략하고 문장을 연결한다.
- JO_4: 두 문장 A, B가 주어만 다르고 나머지 부분이 같을 경우 문장 A와 B의 주어를 하나로 합치고 나머지 부분을 넣어줌으로써 두 문장을 결합한다.
- JO_5: 두 문장 A, B가 동일한 주어를 가지며 다른 동사를 가지지만 두 동사의 의미를 포함할 수 있는 하나의 동사가 존재할 경우, 이 동사를 사용하여 두 문장을 결합한다.

표 1은 문장결합 연산자의 적용 예를 보여주고 있다.

표 1. 문장 결합 연산자의 정의

연산자	문장 A	문장 B
	결합 문장	
JO_1	그는 도착지1에 간다.	그녀는 도착지2에 간다.
	그는 도착지1에 가고 그녀는 도착지2에 갑니다.	
JO_2	그는 사물1을 좋아한다.	그는 사물2를 싫어한다.
	그는 사물1을 좋아하고 사물2를 싫어한다.	
JO_3	그는 시간에 떠난다.	그는 출발지를 떠난다.
	그는 시간에 출발지를 떠난다.	
JO_4	그는 도착지에 간다.	그녀는 도착지에 간다.
	그들은 도착지에 간다.	
JO_5	그는 출발지를 떠난다.	그는 도착지에 도착한다.
	그는 출발지에서 도착지로 간다.	

3.2 문장계획 트리의 해석

문장계획 트리는 도메인 정보 삽입과정과 문장 골격 해석과정을 통해서 하나의 문장을 생성한다. 도메인 정보 삽입과정에서는 말단 노드의 템플릿 변수를 적절한 도메인 정보로 치환하여 단문을 만든다. 만일 이때 해당 템플릿 변수의 도메인 정보가 없다면, 말단 노드의 템플릿은 해당 도메인 정보를 묻는 질문형의 문장이 된다. 그리고 문장 골격 해석과정에서는 문장결합 연산자의 정의에 따라 문장을 결합하여 하나의 복문을 생성한다. 그림 2는 문장계획 트리의 해석과정을 보여주고 있다.

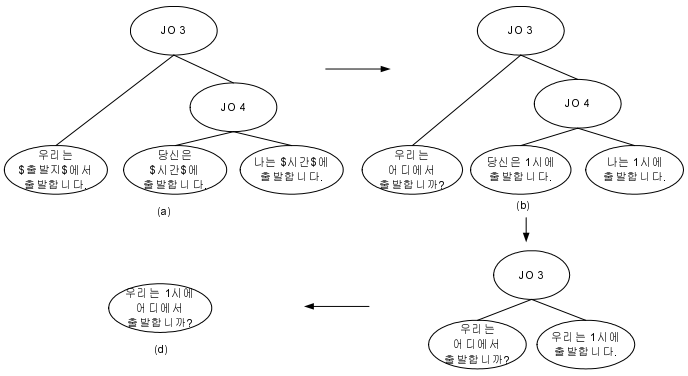

그림 2. 문장계획 트리의 해석과정

3.3 문장계획 트리의 진화

그림 1에서 보듯이 문장계획 트리로 이루어진 유전자 트리들은 유전자 프로그래밍의 교차연산자와 돌연변이 연산자를 통해서 다양한 개체들로 생성되며 선택 연산자를 통하여 사용자로부터 높은 점수를 얻은 개체들이 다음 세대에 살아남게 된다. 유전자 트리의 교차, 동련변이 연산 및 생성은 자연스러운 문장의 표현을 위해, 말단 노드에 동일한 도메인 정보를 갖는 템플릿이 존재하지 않도록 하며, 트리의 말단 노드의 수는 너무 길고 부자연스러운 문장이 생성되지 않도록 n개로 제한하였다. 그림 3과 4는 문장계획 트리의 교차와 돌연변이 연산의 예를 보여주고 있다.

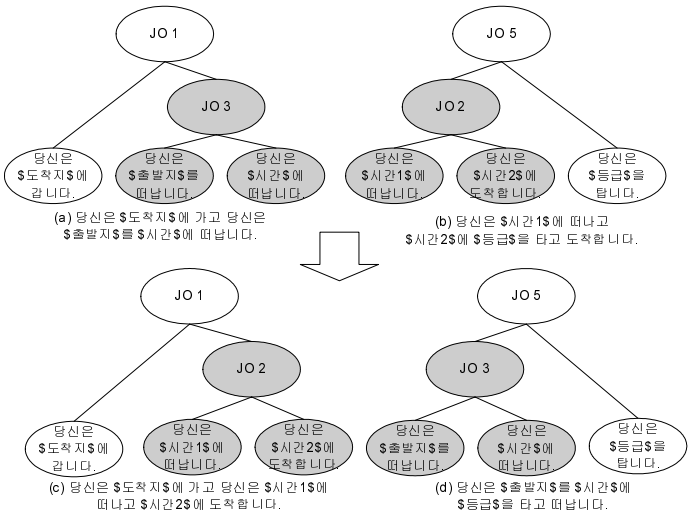

그림 3. 교차 연산자

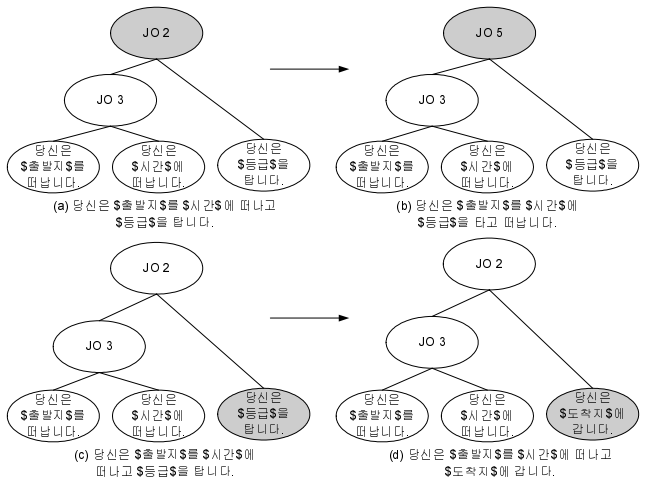

그림 4. 돌연변이 연산자

4. 실험 및 결과

본 논문에서는 제안하는 방법의 유용성을 평가하기 위해서 문장생성 기법 중 가장 널리 사용되는 템플릿기반 방법과 제안하는 방법을 대화형 에이전트에 적용하여 설계의 편의성과 문장의 다양성의 관점에서 비교하였다. 실험은 총 12명의 피험자들을 3개의 그룹으로 나누어 수행되었다. 첫 번째 그룹의 목적은 제안하는 방법을 적용한 시스템을 설계하는 것으로 템플릿 설계하고 100 세대까지 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 5는 첫 번째 그룹이 시스템을 설계하는데 걸린 시간을 보여주고, 그림 6은 피험자가 설계한 템플릿의 평균 개수를 보여주고 있다.

템플릿의 설계과정은 초기에는 단문 템플릿을 위주로 빠르게 템플릿을 추가할 수 있지만, 템플릿의 개수가 늘어남에 따라서 템플릿의 중복체크와 복합문장의 자연스러움 등을 고려해야 하므로 템플릿의 추가 속도가 떨어졌다.

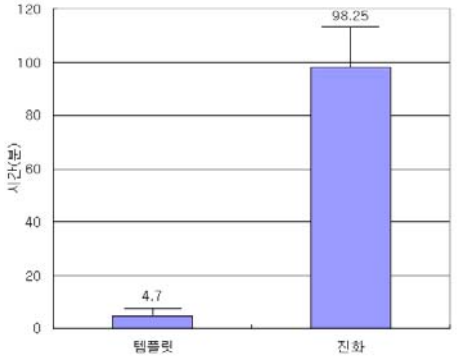

그림 5. 제안하는 방법의 시스템 설계시 드는 시간 비용

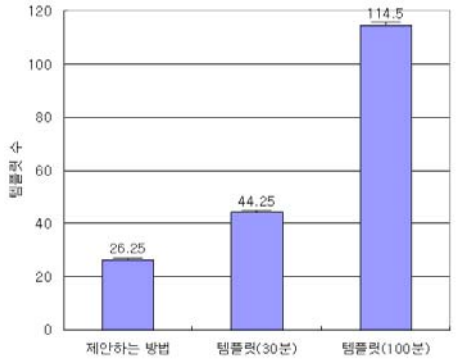

그림 6. 설계된 템플릿의 개수

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. 실험결과(그림 7) 제안하는 방법이 적은 템플릿을 가지고 다양한 문장을 생성할 수 있음을 확인하였다.

5. 결론

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 유전자 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.

감사의 글

이 연구는 산업자원부가 지원한 뇌과학 연구 프로그램에 의해 지원되었음.

참고문헌

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

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

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

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

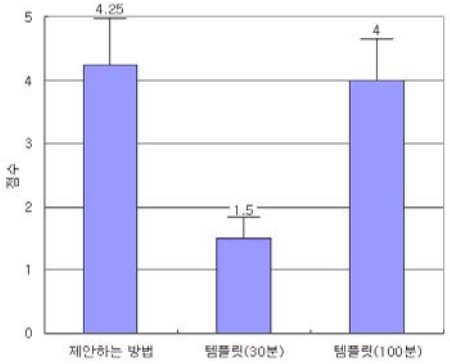

그림 7. 문장의 다양성