한글 기반의 크로스워드퍼즐 자동생성

이승희[O+] 조환규[++]
[+]부산대학교 산업대학원 ,[++]부산대학교 컴퓨터 공학과
{ shlee[O],adagio}@pearl.cs.pusan.ac.kr

Automatic Generation of Crossword Puzzle with Korean Character

Seunghee Lee[O+] Hwan-Gue Cho[++]
[+]Dept. of Computer Engineering, Graduate School of Industry, Pusan Nation University
[++]Dept. of Computer Science, Pusan National University

요 약

크로스워드 자동 생성 시스템은 대량의 한글 낱말을 대상으로 하여 가로 세로 10 정도의 크기를 지닌 크로스워드 퍼즐을 자동으로 만들어주는 시스템이다. 크로스워드 퍼즐은 신문이나 잡지 등에서 많이 볼 수 있으며 이를 자동화해서 생성해주는 프로그램들이 많이 개발되어있다. 그러나 지금까지 개발된 자동 시스템들은 영어를 대상으로 하는 프로그램들이 대부분이며, 한글을 대상으로 하는 시스템은 거의 없는 실정이다. 본 논문에서는 한글 기반의 크로스워드 퍼즐을 자동으로 생성하기 위해 한글 사전의 특성을 조사하여 보고 크로스워드 퍼즐 자동 생성 시 노드들의 연결을 그래프 구조로 변환하여 생각을 해보았다. 그리하여 그래프에서의 사이클 허용여부와 노드가 가지는 connectivity를 크로스워드 퍼즐 자동 생성 시스템에 적용을 하여 보았다.

1. 서 론

크로스워드 퍼즐은 낱말 맞추기 게임의 하나로서 글자가 들어갈 흰색의 빈칸과 검은색 칸들로 구성된다. 번호가 매겨진 빈칸에 들어갈 가로 세로 글자의 힌트나 뜻을 보고 빈 칸 하나에 낱글자 한 개를 집어넣어 빈칸을 채우는 방식의 게임이다. 각 낱말들은 한 두 곳에서 다른 낱말들과 서로 교차하는데 여기에서 크로스워드 퍼즐이라고 이름을 붙였다. 자동 프로그램들이 개발되기 전에는 사람이 직접 크로스워드 퍼즐의 칸들을 적절히 배치하고, 들어갈 낱말들을 찾아 뜻을 나열하는 수동의 방식으로 퍼즐 문제를 만들었다. 이 과정은 매우 복잡하고 일일이 사전을 뒤져서 교차하는 글자들을 다 맞추어야 하는 번거로운 일이다. 이런 문제를 해결하기 위해서 사전에 수록된 단어들을 바탕으로 하여 자동으로 퍼즐을 생성해주는 프로그램들이 많이 개발되어있다. 알려진 것들로는 Crossword Compiler [1], Crossword Constructor[2] 등이 있다. 하지만, 지금까지 개발된 자동 시스템들은 대부분 영어나, 불어와 같은 언어를 대상으로 한 것으로, 한글 기반의 시스템은 거의 없는 실정이다. 크로스워드 퍼즐이 적용되는 분야의 사전만 바꾸어주고서 자동으로 퍼즐을 생성할 수 있다면 여러 면에서 아주 유용하게 쓰일 수 있다.

본 논문에서는 한글 기반의 크로스워드 퍼즐을 자동으로 생성할 때 한글의 특징이 반영되어야 하는 이유를 사전 분석을 통해 설명하고, 낱말을 검색하기 위해 만든 사전에 대해 간단히 소개를 한다. 그리고 자동 생성을 위해 고려해야하는 몇 가지 사항들을 살펴보고 마지막으로 본 논문에서 제시하는 방법을 바탕으로 제작된 시스템에 대해서 간단히 소개하겠다.

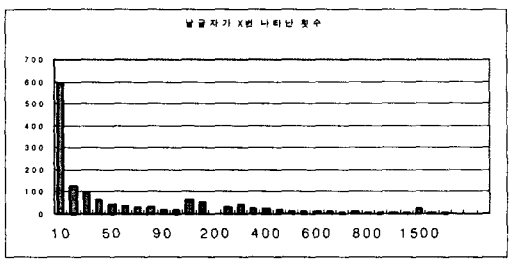
2. 본 론

2.1 한글의 특수성

64,000 여개의 글자를 지닌 한글 사전을 대상으로 하여 다음과 같은 사항들을 조사하여 보았다. 낱말들의 평균길이, 길이별 분포, 낱글자의 개수, 낱글자들의 빈도수, 위치 p에서 볼 수 있는 낱글자의 수 등 여러 가지로 살펴본 결과 다음과 같은 결과를 얻을 수 있었다.

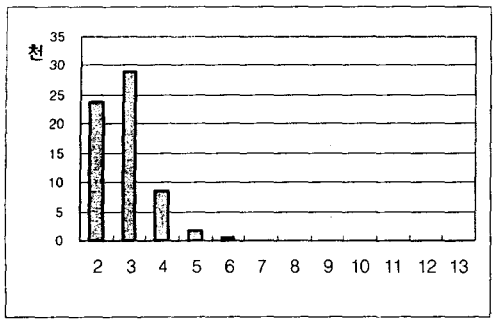
① 영어에 비해 낱글자들의 종류가 많다.
- 영어는 낱글자가 26 개의 알파벳 밖에 없다.
- 한글은 표준 자음 19개와 표준 모음 21개, 이중 모음 11개가 초성, 중성, 종성의 결합으로 글자를 구성한다. 이때 생길 수 있는 가능한 낱글자의 수는 11,172 개이다. 그 중 크로스워드 퍼즐에서 쓸 수 있는, 길이가 2 이상인 낱말들에서 나타나는 낱글자들의 총 수는 1,403 개가 쓰이고 있다.

② 낱글자들의 빈도가 고르지 않다.
- 예를 들면 '굉(1회)','밖(5회)'처럼 아주 적은 빈도로 나타나는 낱글자도 있는 반면 '이(3068회)','기(2725회)' 처럼 아주 많은 빈도로 나타나는 글자들도 있다. [그림1]을 보면 전체 사전에서 10번 이내로 나타난 낱글자들의 수가 591 개 가 된다. 1403 개 낱글자 중의 42%가 10번 이내로 나타나는 글자들인 것이다.



[그림 1] 낱글자들의 등장 횟수

③ 전체 낱말의 80%이상이 길이 2~3 에 몰려있다.
- 길이별 분포를 조사해 보면 전체의 95% 이상이 길이 4 이하이다. [그림 2]에서 볼 수 있다.



[그림 2] 길이별 분포

2.2 이전 연구

현재 crossword 퍼즐에 대한 논문은 비교적 적다. 또 영어를 사용하는 퍼즐은 규칙이나 퍼즐 판의 모양에 대한 규정들이 다양하여 논문들의 종류도 다양한 부분들을 다루고 있다.[3] 그 중 주목할 만한 두 개의 논문에서 Puzzle World를 구축하는데 필요한 몇 가지 사항을 정리해보겠다. M.L. Ginsberg[4]은 24000단어 이상, 4 X 4 이상의 퍼즐에 대해 word-by-word 기법으로 search를 하고 있다. 그리고 Peter Gary[5]는 기존의 논문들을 정리하여 노드의 형태를 미리 결정한 경우는 constrained, 그렇지 않은 경우는 unconstrained로 구분을 하고 있다. 또 노드에서 퍼즐을 채워나갈 때 적당한 글자가 없어서 실패를 할 경우를 위하여 후보자를 두고 있다. 즉, 실패가 일어나면 전 단계에서 확장이 일어났던 지점으로 돌아가 다른 후보자를 검색하면 되도록 하였다. 이 과정은 tree의 탐색기법 중 DFS 기법과 같다.

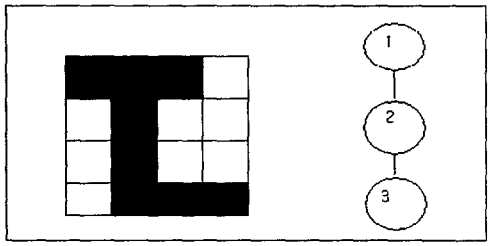
이 두 논문의 내용 중 후자의 논문에서 언급이 된 constrained/unconstrained 개념과 후보자의 개념을 본 시스템에서도 적용할 것이다. unconstrained 의 경우는 자동화의 전 단계로서 사용자가 노드의 형태들을 결정하면 unconstrained 조건을 지닌 시스템이 맞는 글자들을 찾아서 퍼즐 문제를 작성하여 준다. constrained 의 경우는 완전 자동화 생성 시스템으로서 퍼즐 판의 복잡도와 후보자의 수에 따라 결과들을 보여준다.

2.3 시스템에서 고려해야 될 문제들

자동 생성 시스템에서 변화를 줄 수 있는 요인들은 퍼즐 판의 크기, 한 개의 노드가 가질 수 있는 자식의 수(connectivity), 실패를 할 경우 backtracking을 위한 후보자의 수 등이다.

2.3.1 퍼즐 판의 크기

퍼즐 판의 크기는 퍼즐 판의 노드가 가질 수 있는 최대 개수와 연관이 있다. 이것은 퍼즐 판을 tree 구조로 여길 경우 tree의 깊이와 연관이 있다. 보통 볼 수 있는 퍼즐의 크기가 10 X 10 이므로 여기서도 10 X 10 까지 가능하도록 하였다.



[그림 3] 퍼즐 판의 노드와 tree의 연관관계

2.3.2 크로스워드 퍼즐의 복잡도

이 시스템에서는 복잡도를 한 개의 노드가 가질 수 있는 자식의 수로 제안하였다. 부모노드에서 자식 노드로 확장을 할 때 허용할 수 있는 수를 n 이라 하면 부모 노드가 갖는 낱말의 길이 L(>=2) 은 2n - 1의 길이를 지녀야 한다. n=3 까지 확장을 허용한다고 하면 낱말의 길이가 5이상이어야 한다. 그러나 길이 5 이상인 글자는 전체 사전의 단어들 중 2% 밖에 되지 않는다. 그러므로 퍼즐 문제를 구성하는 데 실패할 확률이 높다. 그러므로 확장 가능한 수를 2까지만 허용을 한다.

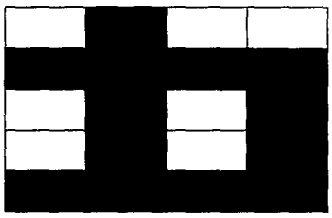
만약 n=1 이라면 전체 노드들이 자식을 한 명만 두는 skewed tree 의 형태를 갖게 되고 [그림2], n=2 라면 binary tree 가 된다. n=1.5 라면 퍼즐 판의 전체 노드 개수 중 절반 정도가 자식을 2 가질 것이다.

2.3.3 Backtracking

다음으로 확장할 적합한 글자를 찾지 못했을 경우 backtracking이 필요하다. backtracking은 바로 전 노드로 돌아가서 놓아 놓은 k 개의 후보자 중 다른 하나를 선택해서 다시 행해나가면 된다.

2.3.4 사이클 허용 여부

마지막으로 고려를 해야 하는 한 가지 사항이 있다. 노드의 모양을 결정해가는 과정에서 사이클을 허용할 것인지 아닌지를 고려해보아야 한다. 사이클은 [그림3]같은 상태를 말한다.



[그림 3] 사이클이 생긴 경우

퍼즐 문제를 만들 때 두 개의 낱글자 x1,x2가 들어 있으면서 두 낱글자들 간의 거리가 정해져 있는 조건을 만족하는 낱글자를 찾아야 한다. 전체 사전에서 이런 글자들을 만날 수 있는 확률이 높다면 사이클을 허용하여도 되지만 확률이 낮다면 실

[표 1] 낱글자 쌍이 들어있는 글자의 개수별 분포

개수	조합수	개수	조합수	개수	조합수	개수	조합수
1	18962	16	41	31	6	48	1
2	5792	17	30	32	2	50	1
3	2618	18	34	33	5	51	1
4	1364	19	24	34	4	56	1
5	908	20	25	35	1	59	2
6	562	21	21	37	1	64	1
7	338	22	25	38	1	68	2
8	289	23	12	39	3	70	1
9	211	24	10	40	1	71	1
10	141	25	8	41	1	75	1
11	138	26	11	42	2	91	1
12	102	27	7	43	1	96	1
13	68	28	7	44	1		
14	58	29	6	46	2		
15	43	30	6	47	2		

패의 확률이 높기 때문에 허용을 할 수 없다. 인명과 외래어가 다 포함이 되어있는 64,000 개의 글자를 가진 사전에서 임의의 두 낱글자 쌍에 대해 사이클에 적용할 수 있는 경우의 수를 조사해 보았다. 가능한 낱글자의 조합의 수 190만 개 중 실제로 크로스워드 퍼즐에 쓰일 수 있는 낱글자의 쌍(거리 2 이상)은 31,908 개 이다.

[표1]은 낱글자 x1,x2 가 속해있는 임의의 낱말을 골랐을 때 해당 낱말이 하나밖에 없는 경우가 18,962 개라는 것을 말해주고 있다.

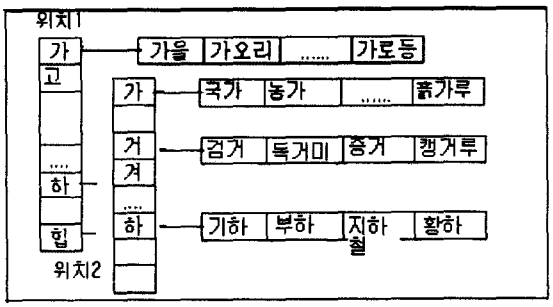
낱글자들의 가능한 조합 전체 쌍 190 만개 중 임의로 낱글자 두개를 골랐을 때 해당되는 낱말이 존재할 경우가 31,908 개로서 한번에 성공할 확률이 1.6 % 밖에 되지 않음을 보여주고 있다. 그러므로 한글 자동 생성 시스템에서는 사이클을 허용하지 않는 것이 바람직하다.

2.4 자동 생성 시스템

2.4.1 사전의 구축

영어와 같은 언어를 다루는 퍼즐은 낱말들을 저장하는 사전의 구조를 낱말의 길이를 위주로 하여 구성을 한다. 또 한글의 효율적인 사전화를 위해 많은 연구들이 있어왔다.[6]

그러나 크로스워드 퍼즐이 위치 p 에 낱글자 X 가 들어있는 것을 찾는 것이 주요 검색과정이므로 위치를 위주로 하여 구성을 하였다. 위치 p에서 볼 수 있는 모든 낱글자 등을 헤드로 하고 여기에 위치 p 에 낱글자 x 가 들어있는 모든 단어들의 list 가 연결된다. 낱말들, 낱글자들, 그리고 뜻은 별도의 배열과 해쉬 테이블에 저장이 되어서 필요에 따라 인덱싱을 할 수 있다.



[그림 4] 본 시스템에서 제안한 사전의 구조

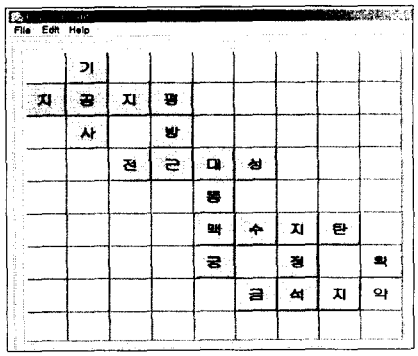
2.4.2 자동화 과정

기존의 방법들은 퍼즐 판의 노드 모양이 미리 결정이 된 상태에서 낱말들을 검색하는 방법을 취하고 있다. 그러나 여기서는 글자 x 를 선택한 뒤, 그 글자에서부터 복잡도 등을 고려하여 다음 글자들을 찾아가는 방법을 쓸 것이다. 자동으로 구성을 하는 것인 만큼 노드의 모양을 미리 결정하는 시간을 없애기 위한 것이다.

① 랜덤하게 퍼즐 판에서 시작 위치와 노드의 길이를 결정.
② 해당하는 글자들을 검색한 뒤 그 중 k개의 후보자를 결정.
③ k 개 중 한 개를 노드에 넣을 글자로 결정한 뒤 어느 위치의 글자에서 확장을 할 것인지 결정을 한다.
④ 해당 위치를 기준으로 해서 다시 적절한 노드의 위치와 길이를 결정한 뒤 ②로 돌아간다.

2.4.3 자동생성 프로그램

[그림4]는 현재 부분적으로 완성된 자동 생성 프로그램의 실행 모습이다. 앞에서 논의한 사전의 구조와 방법으로 복잡도 1의 실행화면이다.



[그림 5] 실행화면

3. 결론 및 향후 과제

한글기반의 크로스워드 퍼즐은 영어에 비해서 많은 낱글자 수와 고르지 않은 낱글자들의 분포, 80% 이상의 낱말이 길이 3이하인 특징들이 있다. 그래서 사이클을 허용하지 않고, 퍼즐 판의 노드에서 다음 노드로 확장 할 수 있는 복잡도도 2 이하로 제한하였다. 여기서 한 가지 주목할 사항은 시스템에서 제안하고 있는 복잡도는 사람이 실제로 퍼즐을 풀 때의 복잡도와는 비례하지 않는다는 점이다. 오히려 시스템이 어렵게 만든 것일수록 글자를 찾을 때의 실마리가 많이 제공이 되므로 오히려 더 쉬울 수도 있다.

그리고 퍼즐 문제를 만들 때 위치 p 에 낱글자 x 가 있는 낱말을 검색하므로 위치를 기준으로 하는 사전을 구축하였다.

앞으로 해결해야 할 과제로는 사용자 인터페이스 부분의 보완이 필요하고, 현재로서는 랜덤하게 노드의 길이나 backtracking을 위한 후보자 선택을 하고 있는데 이 부분에 더 적절한 검색과 결정방법이 연구하여 적용되어야 한다.

4. 참고 문헌

[1] http://www.crossword-compiler.com/
[2] http://www.crosswordkit.com/
[3] J.J.H.Forster et al, "The Crozzle-A Problem for Automation," Proceedings of the 1992 ACM / SIGAPP Symposium on Applied computing: technological challenges of the 1990's, p.110-115, April 1992.
[4] M.L.Ginsberg et al., "Search Lessons Learned From Crossword Puzzles," TR Dept. of CS, Standford Univ., pp. 210-215,1990.
[5] Gary Meehan, Peter Gray, "Constructing Crossword Grids: use of Heuristics vs Constraints," SGES Publications, British Computer Society, pp 159-174, 1997.
[6] 이 승선 외 4인, "Compact*TRIE Index(CompTI) : 한국어 전자사전을 위한 데이터베이스 색인 구조," 정보과학회 논문지, pp. 3-12,1995.1
[7] Geoff Harris 외 2인, "Basic Blocks in Unconstrained Crossword Puzzles," ACM Symposium on Applied Computing, pp. 257-262, 1993.