m차 Markov 한글 정보원의 효율적인 구현에 관한 연구

남기동, 홍종준, 김인대, 이균하
인하대학교 전자계산학과

A Study of An Efficiant Implementation for the m-th order Markov Hangul Information Source

Kidong Nam, Jongjoon Hong, Endae Kim and Kyoonha Lee
Department of Computer Science, Inha University

요 약

본 논문은 한글 정보원을 Markov source로 구현하였을 때 요구되는 terra byte 이상의 방대한 기억 공간의 점유를 해결하기 위해, 이에 대한 통계자료를 조사하고 이를 기초로 기억 공간을 줄일 수 있는 방안을 제안하였다. 제안된 방식에 의해 한글 정보원을 천이확률에 따라서 구현시 paged list 구조로 7차 이상의 Markov 한국어 정보원을 수백 Kbyte의 기억 공간으로 구현 할 수 있었다. 그리고, Markov 한국어 정보원의 활용도를 넓히기 위하여 backward Markov 정보원을 제안하였다.

본 연구에서 제안한 방법은 한글 문장에서 손실된 단어의 수정뿐만이 아니라 기타 Markov source를 한글에 적용하는 모든 분야에 기초적인 자료로 활용될 것으로 기대된다.

1. 서론

인간의 시각이나 청각을 통한 특정 패턴의 인식은 불확실할 수도 있는 패턴의 인지 후 과거로부터 유추되는 일반 개념과의 연관을 통해 이루어진다. 이때 일반 개념이라는 것은 경험으로 형성되어 이미 알고 있는 통계적 모집단을 말한다[1]. 인간의 이런 인식 과정을 연구하여 컴퓨터를 통한 정보 처리의 자동화에 응용하게 되었는데, 이런 연구 분야를 패턴 인식이라 한다.

문자 인식은 패턴 인식의 한 분야로 인식 대상을 언어의 물리적 표현인 문자로 한다. 인간의 문자 인식은 인지된 각 문자, 단어 또는 문장에 약속된 의미를 부여하여 이루어 진다. 그리고, 문자가 물리적으로 표현됨으로 인해 발생되는 정보의 손실, 즉 잉크의 번짐이나 모자람 등과 같은 여러 요인에 의해 문자가 일그러지거나 제대로 안 나타나는 경우 전후 문자나 문맥 등의 정보를 이용하여 원래의 의도된 문자를 유추하게 된다. 마찬가지로, 컴퓨터에 의한 문자 인식에서도 인식 기능과 함께 손실된 정보

를 올바로 고쳐줄 수 있는 기능이 첨가되어야 한다. 이런 기능은 문자 인식의 후처리 (post processing) 단계에서 이루어지는 것으로서, 이의 방법으로는 Binary n-gram, Modified Viterbi Algorithm, 그리고 Predictor-Corrector Algorithm 등이 사용되고 있다[2][3]. 이 중 각 단어 내의 임의의 문자가 선행되는 문자의 영향을 많이 받는다는 점을 고려하는 Modified Viterbi Algorithm은 해당 언어의 문맥적 지식을 m차 Markov 정보원으로 구현하여 사용하는 대표적인 방법이다[4].

그런데, 임의의 언어를 m차 Markov 정보원 형태로 다룰 때 요구되는 기억 공간의 양은 대상 언어에 따라 상당한 차이를 보인다. 예를 들어, 영어에서는 조합가능한 단어 요소들이 26개 이므로 이를 m차 Markov 정보원으로 표현할 경우 $(26^1+26^2+\cdots+26^{m+1})$개의 entry가 필요하다. 그렇지만, 한글에 있어서는 단어 요소를 문자로 볼 때 조합가능한 문자의 수는 11,172개나 되므로 필요한 entry 수는 $(11172^1+11172^2+\cdots+11172^{m+1})$개가 되고, 이중 사용빈도가 높은 완성형의 2,350개 문자만을 고려하더라도 필요한 entry 수는 $(2350^1+2350^2+\cdots+2350^{m+1})$개에 달한다. 이때의 entry는 발생 정보의 단위로 쓰인 것으로서, 각 entry를 코드와 빈도수를 위해 4 byte로 가정할 경우, 2차 Markov 정보원을 구현하더라도 영어는 약 75 Kbyte, 조합형 한글은 약 5 Terabyte, 그리고 완성형 한글은 약 52 Gigabyte의 기억 공간이 필요하게 되는데, 이런 계산 결과는 한글을 Markov 정보원으로 구현하는데 따르는 어려움으로써, 이에 대한 실제적인 구현이나 응용에 대한 연구가 미비하였다.

따라서, 본 논문은 m차 Markov 한글 정보원과 관계가 깊은 통계 자료를 조사 하였다. 조사 결과 Markov 정보원의 특징인 조건부 확률을 전제로 하였을 경우 사용가능한 문자의 수에 비해 실제로 쓰이는 문자의 수가 매우 적고, Markov 정보원의 차수가 올라갈수록 실제로 사용되는 문자의 수가 급격히 감소하는 현상에 착안하여 작은 용량의 기억 공간으로도 구현이 가능한 Markov 한글 정보원의 실용적 구현에 대한 연구를 하였다. 그리고, 기존의 Markov 정보원의 활용도를 넓히기 위해 단어의 뒤에서부터 천이 확률을 계산하는 backward Markov 정보원을 제안하였다. 본 논문의 연구 결과는 문서 인식이나 자연어 처리 등과 같이 한글을 m차 Markov 정보원으로 사용하는 모든 분야에 기초적인 자료로 활용될 수 있을 것으로 기대된다.

2. m차 Markov 한글 정보원

확률적인 정보원에는 정보원의 각개 요소의 출현 확률이 앞서 출현한 유한의 요소에 영향을 받는 Markov 정보원이 있다. 즉, q개의 요소를 갖는 임의의 정보원의 형태가 선행되는 m개의 유한 요소에 의존할때 이를 m차 Markov 정보원이라 한다. m차 Markov 정보원은 정보원의 요소가 발생하는 시간 계열(time sequence)을

$$S_{j1},\ S_{j2},\ \cdots,S_{jm-1},\ S_{jm},\ S_i$$

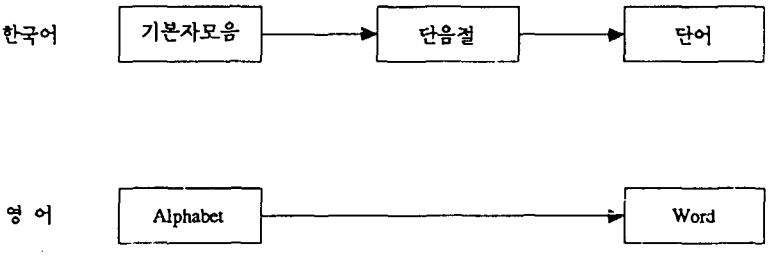
라고 할때, 정보원 요소 S_i가 발생할 조건부 확률은 다음과 같다.

$$P\ (S_i/S_{j1},S_{j2},\cdots,S_{jm-1},S_{jm}) \qquad (1)$$

m차 Markov 정보원은 선행되는 m개의 요소들을 안다면 주어진 요소를 발생시키는 확률을 알수 있는데, 선행되는 m개의 요소들을 m차 Markov 정보원의 state라고 한다.

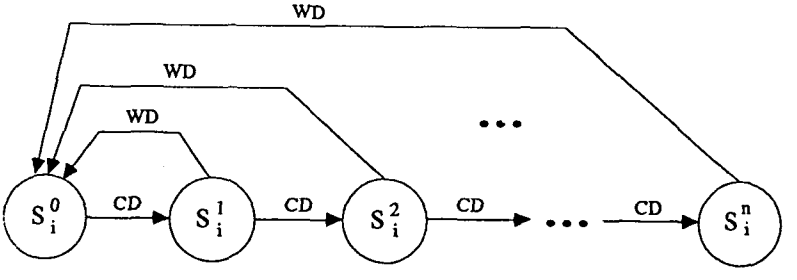
따라서, 정보원이 q개의 요소를 가질때 m차 Markov 정보원은 q^m개의 state를 가져야 한다. 이런 state와 천이 과정을 그림으로 나타낸것이 transition diagram이며, 각 state간의 이동은 천이 확률 P_{ij}에 따라 state i에서 state j로 천이 된다[5][6].

이와 같은 정의를 이용하여 m차 Markov 한글 정보원을 구현하기 위하여 한글 정보원의 특성을 살펴보면 다음과 같다. 한글은 단음절 문자이며, 일정한 규칙에 따라 음에 의해 조합되는 표음 문자이다. 영어에서는 기본 표기 요소인 alphabet의 횡적 나열로서 직접 단어를 구성하지만, 한글에서는 기본 표기 요소들이 2 - 7 개가 초성, 중성 그리고 종성의 순으로 음절 구성 요소를 형성하고, 이들 3성음은 단음절이라는 중간단계를 조직함으로써 영어의 경우와는 매우 다른 과정을 이룬다<그림 1>. 또한 단음절이라는 중간단계에서 생성되는 문자는 모아쓰기를 통해 구성되므로, 사용가능한 전체 문자의 수도 11,172개나 된다.



<그림 1> 한글과 영어의 단어 구성 비교

한글 정보원의 특성을 고려하여, 단어 구성의 요소를 문자들로 가정하여 단어 형성의 transition diagram을 나타내면 <그림 2>와 같다. 이때 S는 state를 나타내며, 위첨자 n은 state 갯수로써 단어가 가지는 문자의 수이다. 아래 첨자 i는 조합가능한



WD : 단어 구분자 (Word Delimeter)
CD : 문자 구분자 (Character Delimeter)
<그림 2> 단어 형성의 transition diagram

문자들 중 사용되는 문자를 나타내며, 가정되는 domain에 따라 i의 범위는 가변적이

다. 즉 조합가능한 한글의 모든 문자를 사용할 경우 1≤i≤11172이 되고, 사용빈도가 높은 완성형 한글을 사용할 경우는 1≤i≤2350이 된다. 문자 구분자(Character Delimeter)는 임의의 문자에서 다음 문자로의 천이를 발생시키며, 이때 문자간의 조건부 확률이 주어진다.

이와 같은 m차 Markov 한글 정보원은 여러가지 요인에 의해 손실되는 문자의 수정에 사용될 수 있는데, 이런 과정은 문자 인식의 후처리 단계에서 수행된다. 특히 Modified Viterbi Algorithm은 Markov 정보원을 이용하는 대표적인 방법이다. 그러나, 이를 이용하기 위해서는 m차 Markov 정보원의 구현이 필요한데, 영어와는 달리 한글의 경우는 단어의 구성 요소인 문자가 너무 많아 이의 구현이 용이하지 않다. m차 Markov 정보원의 전체 entry 수는 (2)의 식으로 계산되므로, 2차까지의 영어와 한글의 entry 수를 비교해보면 <표 1>과 같다.

$$\sum_{i=0}^{m} q^{i+1} \qquad i : \text{차수} \qquad (2)$$

<표 1> 영어와 한글의 entry 수의 비교

차 수	영 어	한글(조합)	한글(완성)
0 차	26	11172	2350
1 차	698	$0.125*10^{09}$	$5.53*10^{06}$
2 차	18846	$1.394*10^{12}$	$13.0*10^{09}$

<표 1>에서 알 수 있듯이, 영어의 경우 26개의 alphabet으로 단어를 구성하므로, 2차 Markov 정보원으로 구현할 경우 18846개의 entry가 필요하다. 그러나, 한글의 경우는 11,172개의 문자로 단어를 구성하므로 2차 Markov 정보원으로 구현할 경우 $1.394*10^{12}$개의 entry가 필요하고, 사용 빈도가 높은 완성형 한글의 경우는 $13.0*10^{09}$개의 entry가 필요하다. 그리고, 각 entry는 해당되는 문자 코드와 조건부 확률을 포함해야 한다. 따라서, 조건부 확률대신 빈도수를 넣을 경우, 각 문자 코드를 2byte로 하고 빈도수를 2byte로 하여 entry가 차지하는 기억 공간을 4 byte로 가정할 경우 요구되는 기억 공간의 양은 <표 2>와 같다. 이때 빈도수는 65536으로 제한되는데, 이것은 최소한의 빈도수를 나타내기 위함이고 이보다 클때는 더 많은 기억 공간이 요구된다.

<표 2> m차 Markov정보원의 기억 공간의 비교

(단위 : byte)

차 수	영 어	한글(조합)	한글(완성)
0 차	96	44688	9400
1 차	2792	0.600 Giga	22.12 Mega
2 차	75384	5.576 Tera	52.00 Giga

이와 같이 m차 Markov 한글 정보원의 구현은 요구되는 기억공간의 양이 매우 방대하다. 따라서, 현재의 컴퓨터 환경하에서의 구현은 용이하지 않다. 그러나, m차 Markov 정보원의 특징인 조건부 확률을 전제로 하였을 경우 사용가능한 문자의 수에 비해 실제로 쓰이는 문자의 수가 적고, 따라서 쓰이지 않는 문자를 효율적으로 처리할 수 있다면 필요한 기억 공간을 대폭 줄일 수 있다. 그러므로 위의 특성을 고려하면 m차 Markov 한글 정보원의 경제적 구현이 가능할 것이다. 그러나, 이를 위해서는 m차 Markov 한글 정보원의 통계자료가 필요한데, m차 Markov 한글 정보원에 있어서 문자 구성에 대한 음소단위의 통계량[7][8]에 대한 연구는 있었으나, 단어 구성에 따른 문자의 통계량에 대한 연구는 없었다. 따라서, 본 논문에서는 국민학교 국어 교과서의 낱말 잦기조사[9]를 참고로 하여, m차 Markov 한글 정보원의 단어 구성에 대한 통계자료의 조사와 이를 기반으로 m차 Markov 한글 정보원의 경제적 구현 방안을 제시하였다.

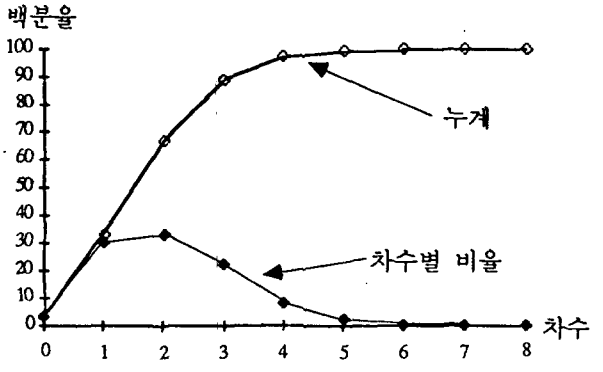
3. m차 Markov 한글 정보원의 경제적 구현 방안

m차 Markov 한글 정보원을 경제적으로 구현하려면, 먼저 이들의 통계자료를 조사하여 정보원의 특성을 파악한 후 이를 근거로 경제적인 구현 방안을 마련하여야 한다. 따라서 본 논문에서는 이런 통계자료를 구하기 위해 국민학교 국어 교과서의 단어들을 사용하였다. 이 단어들은 한글 학회에서 국민학교 국어 교과서의 단어들에 대해 빈도수를 조사한 문서[9]로, 이 자료에 의한 m차 Markov 한글 정보원의 통계자료는 <표 3>과 같다.

0차, 즉 단어의 시작 문자는 1,052로 완성형 한글과 비교해 볼 때 사용가능한 문자중 약 45%가 쓰였음을 알 수 있다. 이는 1차에서 필요한 table의 수가 1,052임을 나타내기도 한다. 이론적으로는 2350개의 table과 2350^2개의 entry가 필요하지만, 실제는 1052개의 table과 9631개의 entry가 발생한것을 <표 3>에서 알 수 있다. 또한 1052개의 table이 포함하고 있는 entry 수도 1개에서 135개까지로 다양하게 나타났다. 이는 각 table이 2350개의 entry를 포함할 수 있는데 비해 실제 발생한 entry는 135개란 점으로 미루어 각 table을 효율적으로 구현한다면 상당히 적은 기억 공간으로 m차 Markov 한글 정보원을 구현할 수 있을 것이다.

〈표 3〉 Markov 한글 정보원의 차수별 entry 수

차 수	이론 entry 수	Entry 수	비율(%)
0 차	2350^1	1,052	3.34
1 차	2350^2	9,631	30.01
2 차	2350^3	10,696	33.28
3 차	2350^4	7,146	22.25
4 차	2350^5	2,596	8.09
5 차	2350^6	744	2.32
6 차	2350^7	204	0.64
7 차	2350^8	20	0.06
8 차	2350^9	3	0.01
계	$\sum_{i=0}^{8} 2350^{i+1}$	32,092	100.00

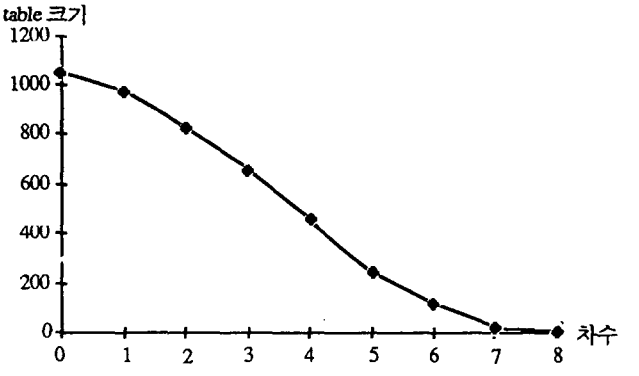


〈그림 4〉 Markov 한글 정보원의 차수별 entry 분포도

조사된 최고 차수는 8차로 "끄덕끄덕하였읍니다","어둑어둑하였읍니다", 그리고 "흐뭇해지시겠읍니다"의 3개 단어였다. 〈표 3〉에서 8차의 entry 수가 단 3개가 발생했다는 것은 이론상의 table이 2350^9개의 entry를 포함할 수 있는데 비해 극소이므로 기억 공간을 대폭 줄일 수 있다. 결론적으로 m차 Markov 한글 정보원은 3차 이상이 되면 실제 발생하는 entry 수는 차수에 반비례한다는 것을 알 수 있다.

〈표 4〉 차수별 table 갯수

차 수	table의 수
0 차	1052
1 차	972
2 차	826
3 차	657
4 차	453
5 차	245
6 차	115
7 차	17
8 차	3
계	4,680



〈그림 5〉 table 갯수의 분포도

〈표 4〉와 〈그림 5〉는 차수별 table의 갯수를 나타낸 것이다. 각 table은 2350의 entry를 포함할 수 있으나, 실제 포함하고 있는 entry의 수는 10%미만으로 나타났다. 그러므로 각 table은 2350개의 entry에 대한 영역을 다 갖기보다는 현재 발생한 entry의 수와 미래에 발생할지도 모를 entry에 대한 영역을 적절이 할당한다면 수백 kbyte로도 m차 Markov 한글 정보원을 구현할 수 있다.

한편, m차 Markov 정보원이 시간 계열하에서 선행되는 m개의 요소에 영향을 받는 특성을 갖는다고 할 때, 시간 계열상의 역과정을 거쳐 정의되는 m차 Markov 정보원을 정의할 수 있다. 이를 m차 backward Markov 정보원이라 가정하자. 그러면 m차 backward Markov 정보원을 다음과 같이 정의할 수 있다. m차 Markov 정보원의 시간 계열이 다음과 같을때

$$S_{j1}, S_{j2}, \cdots, S_{ji-1}, S_{ji}, S_{ji+1}, \cdots, S_{ji+m-1}, S_{ji+m}$$

m차 backward Markov 정보원에서 S_i가 발생할 조건부 확률은 다음과 같다.

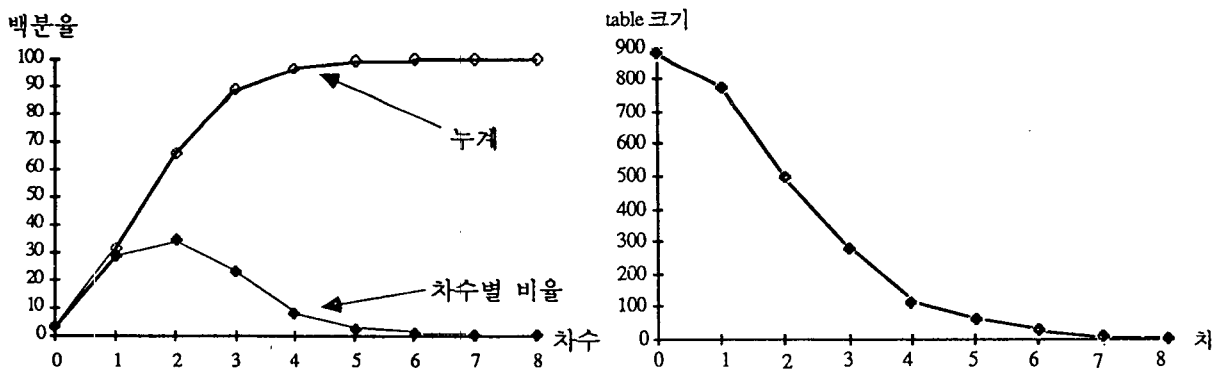
$$P(S_{ji}/S_{ji+m}, S_{ji+m-1}, \cdots, S_{ji+1}) \qquad (3)$$

이런 m차 backwrd Markov 정보원의 사용은 기존의 Markov 정보원이 시간 계열하에서 천이확률을 계산하던 것을 뒤에서부터 천이확률을 계산하므로 활용도를 증대시킬 수 있다. 이런 점에 착안하여 〈표 3〉의 통계자료를 위해 사용하였던 문서를 재구성하여 m차 backward Markov 한글 정보원의 통계 자료를 조사하였다. 〈표 3〉은 m차 backward Markov 한글 정보원의 통계자료로 entry 수는 m차 Markov 한글 정보원과 유사함을 알 수 있다. 그러나, table의 수는 매우 적음을 알 수 있는데, 이는 각 table이 가지는 entry 수가 backward Markov 한글 정보원이 Markov 한글 정보원보다 많음을 나타낸다.

<표 5> m차 backward Markov 한글 정보원의

통계자료

차 수	이론 entry 수	Entry 수	table 수
0 차	2350^1	880	880
1 차	2350^2	8,825	775
2 차	2350^3	10,562	495
3 차	2350^4	7,072	280
4 차	2350^5	2,497	114
5 차	2350^6	749	62
6 차	2350^7	205	26
7 차	2350^8	20	6
8 차	2350^9	3	3
계	$\sum_{i=0}^{8} 2350^{i+1}$	30,183	2681



<그림 6> backward Markov 한글 정보원의 entry 수와 table 수의 분포도

이와 같이 m차 Markov 한글 정보원은 사용가능한 문자에 비해 실제 발생하는 문자의 수가 매우 적었다. 특히 4차 이상이 되면 문자의 수는 급격히 감소함을 볼 수 있

다. 각 entry를 4byte로 가정할 경우 이론상 요구되었던 2차까지의 기억 공간은 52 giga byte였으나, 조사한 통계자료에 따르면 전체 entry가 32,092뿐이 안되므로 130 kbyte로도 8차까지의 구현이 가능하다. 여기에서는 정보원이 국민학교 국어 교과서라는 한정된 자료이나 확대된 자료들을 고려하더라도 수백 kbyte로 구현이 가능할 것이라 생각된다. 그리고, 여기에 m차 Markov 한글 정보원의 활용도를 넓히기 위해 정의되었던 m차 backward Markov 한글 정보원도 함께 구현한다 하여도 기억공간의 양은 1 Mbyte내에서 가능하다고 판단된다.

4. 구현 및 결과

3장에서 조사한 한글 정보원의 특성을 이용하여 m차 Markov 한글 정보원을 경제적으로 구현하기 위해 각 table을 일정한 크기의 page로 나누어 저장하는 paged list 구조를 사용하였다. m차 Markov 한글 정보원은 선행되는 m개의 entry의 영향을 받는 특성을 이용한 것으로, 이의 응용은 여러개의 entry중 가장 영향을 많이 받는 entry를 선택하는 것이다. 따라서, 각 table이 포함하는 entry들을 발생 빈도순으로 정렬 하여 구현할 경우 entry 검색은 빈도수가 많은 것부터 검색을 하므로 이진 검색보다 더 좋은 효율을 가진다. 그러나, 모든 entry들을 순차적으로 구현한다면 미래에 발생 가능한 새로운 정보에 대한 유연성이 떨어지게 된다. 따라서, 본 연구에서는 각 table을 일정한 page로 나누어 순차 편성의 단점을 보완하였다. 그러나, 각 table을 일정 크기의 page로 나누어 구현할 경우 entry의 수가 적은 table은 기억 공간의 낭비를 초래한다. 또한 이런 경향은 차수마다 다르게 나타나므로, 각 차수마다 table이 가지는 entry 수를 조사하여 적절한 page의 크기를 조절하는 것이 필요하다. <표 6>과 <표 7>은 차수별 entry 수와 table 수를 이용하여 page 크기를 결정한 것이다. Page 크기는 차수별 entry 수를 차수별 table 수로 나누어 각 table 당 가지는 평균 entry 수로 정하였다. 이것은 식 (4)와 같이 나타난다.

$$\left\lceil \frac{\text{ENTRY}_i \text{ SIZE}}{\text{TABLE}_i \text{ SIZE}} \right\rceil + 1 \qquad i: \text{차수} \qquad (4)$$

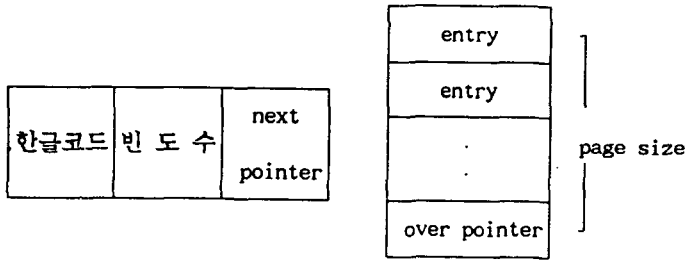
Entry의 자료 구조는 한글코드, 빈도수, 그리고 다음 차수를 지정하는 pointer로서 6 byte로 구성된다. 각 Page는 <표 6>의 차수별 page 크기에 따라 entry를 포함한다. Page의 마지막 entry 부분은 다음 page를 지정하기 위한 pointer이다<그림 7>. 이와 같은 자료구조로 m차 Markov 한글 정보원을 구축하였을 때의 기억공간은 약 330 kbyte로, 단순히 순차편성을 하였을 때보다는 30% 정도의 기억공간이 증가하였다. 증가된 기억공간은 앞으로 발생가능한 새로운 정보에 대한 기억공간으로 볼 수 있다.

<표 6> m차 Markov 한글 정보원의 page 크기

차 수	entry 수	table 수	page 크기
0 차	2,350	2350	2350
1 차	9,631	972	11
2 차	10,696	826	14
3 차	7,146	657	12
4 차	2,596	453	7
5 차	744	245	4
6 차	204	115	3
7 차	20	17	3
8 차	3	3	2
계	33,380	5638	2406

<표 7> m차 Markov 한글 정보원의 page 수와 기억공간

차 수	page 수	기억 공간
0 차	1	9,400
1 차	1,544	101,904
2 차	1,348	113,232
3 차	1,545	74,808
4 차	640	26,880
5 차	325	6,500
6 차	143	2,574
7 차	17	306
8 차	3	36
계	5566	335,640



<그림 7> entry 및 page 자료 구조

m차 backward Markov 한글 정보원도 <그림 7>의 자료구조를 이용하여 구현하였으며, page 크기와 구현하였을때의 기억공간은 <표 8>,<표 9>와 같다. m차 Markov 한글 정보원과 비교해보면 table의 수가 50% 정도 밖에 안되는데, 이는 m차 backward Markov 한글 정보원이 중복된 글자가 많다는 것을 나타낸다. 그러므로, 차수별 page의 크기는 당연히 커지게 된다. 그러나 실제 구현하였을 때의 기억공간은 m차 Markov 한글 정보원과 같이 약 340 kbyte를 차지한다.

〈표 6〉 m차 backward Markov 한글 정보원의 page 크기

차 수	entry 수	table 수	page 크기
0 차	2,350	2350	2350
1 차	8,825	775	13
2 차	10,562	495	23
3 차	7,072	280	27
4 차	2,497	114	23
5 차	749	62	14
6 차	205	26	9
7 차	20	6	5
8 차	3	3	2
계	32,283	4111	2466

〈표 8〉 m차 backard Markov 한글 정보원의 page 수와 기억공간

차 수	page 수	기억 공간
0 차	1	9,400
1 차	1,265	98,670
2 차	867	119,646
3 차	509	82,458
4 차	158	21,804
5 차	103	8,652
6 차	42	2,520
7 차	9	270
8 차	3	36
계	2957	343,438

5. 결론

본 논문에서는 m차 Markov 한글 정보원의 구현시에 요구되는 방대한 기억공간의 점유를 해결하기 위해 이에 관련된 통계 자료를 조사 하였다. Markov 정보원의 특성인 조건부 확률을 전제로 할 경우 차수가 올라갈수록 발생하지 않는 문자의 수가 많은점을 고려하여, 이론적으로 필요했던 수 terabyte의 기억 공간을 약 300 kbyte로 줄일 수 있었다. 구현된 Markov 한글 정보원은 8차이며, 가장 많은 분포를 보인 단어들은 3자나 4자로 구성된 것들이었다. 그리고, Markov 정보원이 선행되는 m개의 요소에 영향을 가장 많이 받는 요소를 선택한다는 특성을 효율적으로 이용하기 위해, 빈도수에 따라 정렬된 요소들을 paged list 구조로 구현하였다. 일반 순차 편성보다는 기억공간이 더 차지하지만, 임이의 요소를 선택하는데 있어 빠른 검색과 새로운 자료에 대한 추가가 용이하다. 또한, 임이의 시간 계열에서 정의되는 m차 Markov 정보원을 보다 효율적으로 사용하기 위해서 m차 backward Markov 정보원을 제안하였다.

사용한 문서가 국민학교 국어 교과서인점으로 미루어, 보다 보편성있는 Markov

정보원을 구현하려면 보다 다양한 문서를 통해 보강되어야 한다. 그러나, 본 논문에서 보인 m차 Markov 한글 정보원의 분포는 다양한 문서를 사용하여도 많이 변하지 않을 것으로 추측된다. 이때 적당한 태그 필드(tag field)를 두어 정보원을 전문분야별로 자료를 따로 관리한다면, 전문분야별 특성에 맞는 효과적인 응용이 가능하리라 생각된다.

참고문헌

[1] 오 영환, "패턴 인식론", pp. 63-162, 정익사, 1991

[2] J.J. Hull and S.N. Shihari, "Experiments in Text Recognition with Binary n-Gram and Viterbi Algorithm," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, pp. 520-530, 1982.

[3] R. Shinghal and G.T. Toussaint, "A Bottom-Up and Top-Down Approach to Using Context in Text Recognition," Man-Machine Studies, Vol. 11, pp. 201-212, 1979.

[4] R. Shinhgal and G.T. Toussaint, "Experiments in Recognition with the Modified Viterbi Algorithm," IEEE Pattern Anaysis and Machine Intelligence, Vol. PAMI-1, pp.184-193, 1979.

[5] Norman Abramson, "Information Theory and Coding", pp. 11-40, McGraw-Hill,New York, 1963.

[6] Ronald A. Howard, "Dynamic Probablistic Systems", pp. 1-19. John Wiley & Sons, New York.

[7] 김 경태, 이 균하, "한글 정보의 경제적 처리를 위한 부호화에 관한 연구", 한국 정보과학 회지, Vol.5, No.2, pp. 99-104, Dec 1978.

[8] 최 홍문, "한국어의 Entropy와 Redundancy에 관한 연구", 인하대학교 전자공학과, 석사학위논문, 1973.

[9] 국민학교 국어 교과서 낱말 잦기 조사, 이 광규, 한글 학회, 1981.