통계적 방법에 의한 후처리

박진우, 이일병 연세대학교 전산과학과

Postprocessing with statistical methods

Park Jin Woo, Lee Il Byung, Yonsei Univ. Computer Science Dept.

본 논문에서는 통계적 방법을 이용한 후처리기를 설계하고, 구현하여 평가하였다. 통계적인 방법은 처리 속도보다는 공간 효율을 높임으로써 후처리의 성능을 높일 수 있다는 가정에서, 후처리의 성능을 향상시키기 위해서 다음의 3가지 방법를 제안한다. 첫째, 전이 확률에서 중복 표현되는 정보를 정의하고, 제거할 수 있는 방법을 제안한다. 둘째, 정수인 순위값으로부터 실수인 전이 확률의 근사값을 추정해 냄으로써 공간 효율을 높일 수 있는 방법을 제안한다. 셋째, 위의 두가지 방법을 복합적으로 적용하여 공간 효율을 높은 오류 탐지와 오류 교정 방법을 제안한다.

제 1 장 서론

1.1 연구 동기

후처리란 텍스트(text)을 컴퓨터에 융통성(flexibility)있게 입력할 수 있도록 하기 위해서 부가적으로 수행하는 작업이다. 융통성이 있다는 것은 문자 인식 또는 음성 인식과정에서의 모호성(ambiguity)이나 입력되는 문자열 내부에 존재하는 오류에도 불구하고 올바른 문자열을 찾아낼 수 있음을 의미한다[SRIH 85].

문자열에 중복성이 내재하고 입력 방법에 따라서 발생하는 오류의 특징이 있기 때문에 융통성을 부여할 수 있다. 문자열에서의 중복성이란 문자열 내에서 어떤 문자가 없어도 그 문자 주위의 문자들을 통해서 알 수 있는 경우에 그 문자는 중복성을 띈다고 한다. 일반적으로 영문 텍스트의 경우 70~80퍼센트가 중복되어 있다고 알려져 있다[THOM 91]. 한편 문자열에 내재한 중복성에 대한 정보를 표현한 것을 문맥정보(Contextual Information)라고 한다.

입력에 융통성을 부여하는 기술은 철자검사 및 교정 뿐만 아니라, 문서인식 또는 음성인식의 성능을 향상시키기 위한 후처리에서 사용될 수 있다.

인간이 문자을 인식할 경우에는 인식할 문자의 양이 증가할수록 더 잘 인식하는 반면에 날자 문자 인식기의 경우에는 문자의 양이 증가할수록 오인식되는 문자의 수는 누적되어 증가한다. 이처럼 인간의 인식 능력과 날자 문자 인식기의 능력 사이에는 질적인 차이가 있다. 이런 인간의 뛰어난 인식 능력은 문자의 순서, 단어의 의존성, 문장의 구조, 통사구조, 문체 뿐만 아니라, 글의 주제 및 내용 등과 같은 문자열 내의 문맥정보를 자연스럽게 활용하기 때문에 가능한 것이다[HULL 82], [SRIH 83]. 따라서 문자 혹은 음성 인식기 능력의 한계를 극복하기 위해서는 문맥정보를 활용해야한다.

1.2 연구 배경

문맥정보는 표상하는 방법에 따라서 구조적 방법, 통계적 방법 그리고 그 두가지를 복합적으로 사용하는 방법으로 나누어진다[SHIN 79], [SRIH 83], [민병우 91]. 구조적 방법은 단어 사전, 문장 구조, 의미망(semantic network)으로써 문맥정보를 표현하므로 하향적인(Top-down) 접근 방법 또는 개념 주도 알고리즘(Concept Driven Algorithm)이라고 하며, 그 접근 방법을 사용한 예는 [BLED 66], [DUDA 68], [ABE 71], [WAGN 74], [OKUD 76], [OOMM 87], [TANA 87], [강재우 89], [박종만 90], [채영숙 92]가 있다. 통계적 방법은 문자들간의 통계적인 분포로 문맥정보를 표현하므로 상향적인(Bottom-up) 접근 방법 또는 데이타 주도 알고리즘(Data Driven Algorithm)이라고 하며, 그 접근 방법을 사용한 예는 [RAVI 67], [FORN 73], [ULLM 77], [TOUS 78], [HALL 80], [KUHN 90], [ARAK 93]가 있다. 그리고 복합적 방법은 통계적 방법의 빠른 처

리 속도 및 교정 후보 생성 능력을 취하고, 구조적 방법의 높은 교정율을 취함으로써 성능을 개선하려는 방법으로 그 예로는 [SHIN 79], [SRIH 83], [KONN 93], [이현일 93], [LEE 93b], [황호정 94]이 있다.

1.3 연구 방향

구조적 방법은 형태소 분석을 선행해야 하는데, 형태소 분석은 자연어처리의 근본적인 문제인 중의성과 얽혀있으므로 현재로는 쉽게 해결되지 않는다. 일반적인 문장의 말마디를 분석할 때, 약 13%의 말마디가 2가지 이상으로 분리되는 중의성을 갖는다[김병희 93]. 복합적 방법도 같은 이유로 한글에서는 어려움이 있다. 반면에 통계적 방법은 상향적 방법이므로 언어의 구조적 특징에 덜 영향 받는다. 따라서, 본 논문에서는 통계적 방법에 의한 후처리를 연구하고자 한다.

통계적 방법은 처리 속도가 빠른 반면에 교정율이 낮다는 특징이 있다[민병우 91]. 통계적 방법에서 개선되어야 할 부분은 처리 속도라기 보다는 오류교정율이다. 통계적 방법에서 오류교정율은 많은 문맥정보를 교정에 이용함으로써 높일 수 있다[HULL 82]. 왜냐하면 오류 주위의 문자열을 교정후보를 결정하는데 더 많이 이용하면 할수록 더 정확한 판단을 할 수 있기 때문이다.

통계적 방법의 성능을 향상시키기 위해서는 일정한 공간에 얼마나 많은 문맥정보를 효과적으로 저장할 수 있느냐에 달려있다. 따라서 본 연구에서는 기존의 문맥 표현 방법에서 중복 표현되는 정보를 찾아서 정의하고, 이를 제거할 수 있는 자료구조를 제안함으로써 후처리기의 성능을 향상시키고자 한다.

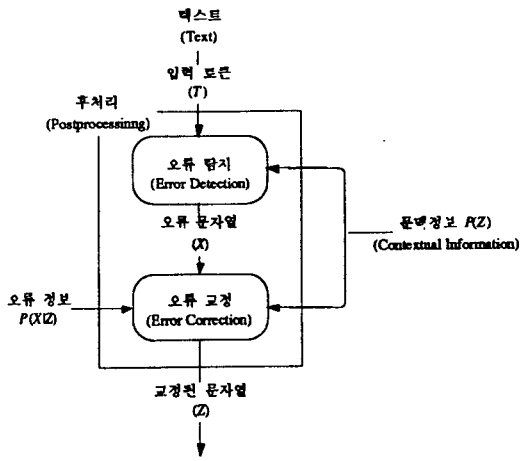
일반적으로 공간 효율과 계산 효율은 절충적인(trade-off) 관계이다. 통계적 방법에서도 예외는 아니다. 중복 표현된 정보를 제거하여 저장하는 방법은 공간 효율은 높지만 계산량이 증가된다. 선형화된 마지널 인덱싱 기법을 개선하여 공간 효율을 높이고, 이 방법에 순위 정보를 이용하여 계산 효율을 향상시킴으로써 좋은 성능을 얻을 수 있는 방법을 찾고자 한다.

제 2 장 통계적 방법에 의한 후처리

이 장에서는 통계적 방법에 의한 후처리의 이론적 배경과 그 성능을 향상시킬 수 있는 점들을 살펴본다.

2.1 이론적 배경

후처리는 오류를 탐지하는 과정과 오류를 교정하는 과정으로 나눌 수 있다. 오류 탐지는 입력 문자열 중에서 오류를 찾아내는 과정이고, 오류 교정은 탐지된 오류를 포함하는 문자열을 입력으로 받아서 올바른 문자열로 바꾸어 주는 과정이다. 〈그림 2.1〉은 통계적 방법으로 후처리기를 설계하였을때에 자료 흐름도(Data Flow Diagram)이다.



〈그림 2.1〉 후처리의 자료 흐름도

본 논문에서 사용하는 기호는 〈표 2.1〉과 같다.

〈표 2.1〉 사용 기호의 정의

$T = T_1 T_2 \cdots T_{n+1}$
: 텍스트에서 분리해낸 길이 $n+1$의 토큰

$X = \overbrace{X_1 \cdots X_n}^{Context} \overbrace{X_{n+1} \cdots X_{n+m}}^{Error} \overbrace{X_{n+1+m} \cdots X_{2n+m}}^{Context}$
: 오류를 포함하는 문자열

$Z = \overbrace{X_1 \cdots X_n}^{Context} \overbrace{Z_{n+1} \cdots Z_{n+m}}^{Correction} \overbrace{X_{n+1+m} \cdots X_{2n+m}}^{Context}$
: X를 교정하여 얻은 문자열

$S = \{S_1, S_2, \cdots, S_r\}$
: 사용 문자 집합 (N-바이트 코드 : r=34)

n : 마르코프 가정의 차수

m : 오류 탐지기가 발견한 오류 문자열의 길이

$T_i \in S, (1 \le i \le n+1);\ X_j \in S,\ Z_j \in S\ (1 \le j \le 2n+m)$

통계적 방법에서 문맥정보는 문자열의 확률분포로 표현된다. 통계적 방법에서 오류 탐지는 $P(T)=0$ 인 T를 찾아내는 것으로 정의되며, 오류 교정은 $P(Z|X)$가 최대값이 되는 Z를 찾아내는 것으로 정의된다. $P(T)$란 입력 토큰 T가 학습 데이타에서 나타날 확률이고, $P(Z|X)$는 오류를 포함하는 문자열 X에 대해서 Z가 입력하려했던 문자열일

확률이다.

오류 탐지는 텍스트에서 길이 $n+1$인 토큰을 입력 받아서, 오류 문자 m개와 그 문자열의 앞뒤에 각각 n개씩의 오류가 아닌 문자열로 구성된 X를 출력한다. 따라서 X의 길이는 $2n+m$이 된다. m개의 오류 문자는 $P(T)=0$ 인 m개의 토큰들간에 공통적으로 포함되는 문자들로 정의된다. $P(T)$는 (1)식으로 표현할 수 있다.

$$P(T) = P(T_1 \cdots T_{n+1})$$
$$= P(T_1) \cdot P(T_2|T_1) \cdots P(T_n|T_1 \cdots T_{n-1}) \cdot P(T_{n+1}|T_1 \cdots T_n) \quad (1)$$

(1)식에서 $P(T_{n+1}|T_1 \cdots T_n)$은 $T_1 \cdots T_n$의 문자열이 나타난 후에 T_{n+1}이 나타날 조건부 확률을 의미하며, $P(T)$를 계산하기 위해서는 Σr^n개의 조건부 확률이 필요하다. 오류 탐지 능력은 n의 크기가 클수록 증가한다. 오류 탐지에서는 $P(T)$가 0인지 여부만으로 오류를 확인하므로, 확률값을 0과 1로 정량화(quantize)한 Binary n-gram방법으로 같은 오류 탐지 성능을 유지하면서, 공간효율을 높일 수 있다[RISE 74], [ULLM 77].

오류 교정은 오류탐지의 결과인 X를 입력으로 받아서 교정된 Z를 출력하는 과정이다. 그 과정은 $P(Z|X)$가 최대값을 갖게 하는 Z를 찾아내는 것이며, $P(Z|X)$는 베이어스의 규칙에 의해 (2)식이 성립한다[HART 83].

$$P(Z|X) = \frac{P(X|Z) \cdot P(Z)}{P(X)} \quad (2)$$

$P(X|Z)$는 문자열 Z가 입력되었을 때 X로 인식될 확률이므로, 오류의 특징을 표현한다. $P(Z)$는 Z의 사전 확률(priori probability)이다. 그리고 $P(X)$는 문자열 X가 관찰될 확률이다. 그러나, $P(X)$는 Z에 독립적이므로, Z를 찾아내는 작업에는 영향을 미치지 않는다. 따라서 (2)식에서 $P(X)$항을 제거한 $P(X|Z) \cdot P(Z)$가 최대값이 되는 Z는 $P(Z|X)$가 최대가 되도록 한다.

$$G(X,Z) = \ln P(X|Z) + \ln P(Z) \quad (3)$$

$P(X|Z) \cdot P(Z)$에 자연 로그 함수($\ln$[주1])를 취한 (3)식의 $G(X,Z)$는 $\ln$이 단조증가 함수(monotonic increasing function)이므로, $P(X|Z) \cdot P(Z)$와 일대일 대응관계를 갖는다. 따라서 $P(Z|X)$가 최대값이 되게 하는 Z는 $G(X,Z)$도 최대값이 되게 하므로, $G(X,Z)$가 최대값이 되게 하는 Z를 찾음으로써 오류교정을 할 수 있다[FORN 73], [SHIN 79], [HULL 82], [민병우 91].

$P(X|Z)$는 조건부 독립을 가정을 통해서 $P(Z)$는 마르코프 가정을 통해서, 임의의 m에 대해서 $P(Z)$와 $P(X|Z)$를 산출할 수 있다. 그리고 $G(Z|X)$가 최대값이 되는 Z는 비터비 알고리즘을 사용해서 빠르게 찾아낼 수 있다[HULL 82].

조건부 독립의 가정

$P(X|Z)$는 어떤 문자열 Z에 대해서 문자열 X로 잘못 입력되었을 확률을 의미한다. X에 포함된 오류 문자의 갯수는 m개이므로, Z를 같은 길이의 X로 잘못 입력할 수 있는 경우의 수는 r^m-1이고, 가능한 X의 경우의 수는 r^m이다. 오류의 길이 m에 대한 모든 $P(X|Z)$를 저장하려면, $r^m(r^m-1)$에 비례하는 기억공간이 필요하므로 현실적이지 않다. m은 오류의 길이이므로 X에 따라서 달라지는 값이므로, 한정시킬 수 없다.

$$\ln P(X|Z) = \sum_{i=1}^{m} \ln P(X_{n+i}|Z_{n+i}) \quad (4)$$

$X_{n+1}X_{n+2} \cdots X_{n+m}$ 사이에 조건부 독립(conditional indepedence)을 가정하면 (4)식이 성립되어 r^2개의 $P(X_i|Z_i)$으로 임의의 $P(X|Z)$를 구할 수 있다. 그러나, (4)식은 문자열을 구성하는 문자들이 서로 독립적이라는 가정에서 유도되었기 때문에 인쇄체 문자인식에서는 성립되나, 필기체 문자인식이나 음성 인식의 경우에는 (4)식은 적합하지 않다. 여기서에서 확률 $P(X_{n+i}|Z_{n+i})$는 원래의 문자 Z_{n+i}가 X_{n+i}로 잘못 입력되었을 확률인 혼동 확률(confusion probability)을 나타낸다. [민병우 91]

n 차 마르코프 가정

$P(Z)$는 문자열 Z가 학습 데이타에서 나타날 확률을 의미한다. 길이가 t인 Z의 가짓수는 r^t개이므로, 모든 길이의 Z의 가짓수는 Σr^t가 된다. 따라서 모든 Z에 대한 $P(Z)$를 저장하려면, 무한한 저

[주1] $\log_e$

장공간이 필요하므로, 모든 Z에 대한 확률을 저장하는 것은 현실적이지 않다.

n 차 마르코프 가정(Markov Assumption)을 통해서 길이가 n 이하인 문자열에 대한 확률만으로써 임의의 길이의 문자열에 대한 확률을 계산해 낼 수 있다. n 차 마르코프 가정이란 어떤 심볼이 나타날 확률은 그 심볼 앞에 나타난 n 개의 심볼들에만 의존하고, 그 이전에 나타난 심볼들과는 독립적임을 의미한다. n 차 마르코프 가정을 수식으로 표현하면 (5)식이 된다[REVU 84].

$$P(Z_t|Z_1Z_2\cdots Z_{t-n}\cdots Z_{t-2}Z_{t-1}) = P(Z_t|Z_{t-n}\cdots Z_{t-2}Z_{t-1}) \quad (5)$$

n 차 마르코프 가정하에 $P(Z)$를 정리하면 (6)식이 성립한다[FORN 73], [SHIN 79], [HULL 82], [민병우 91].

$$\begin{aligned} P(Z) &= P(Z_1Z_2\cdots Z_t) \\ &= P(Z_1|Z_1\cdots Z_{t-1})\cdot P(Z_{t-1}|Z_1\cdots Z_{t-2})\cdots P(Z_2|Z_1)\cdot P(Z_1) \\ &= P(Z_t|Z_{t-n}\cdots Z_{t-1})\cdot P(Z_{t-1}|Z_{t-n-1}\cdots Z_{t-2})\cdots P(Z_2|Z_1)\cdot P(Z_1) \end{aligned} \quad (6)$$

(6)식에서 $P(Z_t|Z_{t-n}\cdots Z_{t-1})$은 $Z_{t-n}\cdots Z_{t-1}$문자열이 나타난 후에 Z_t가 나타날 조건부 확률(conditional probability)을 의미하며, 이 확률을 n 차 전이 확률(transition probability)이라 한다. 따라서 n 차 마르코프 가정하에서는 임의의 $P(Z)$을 n 차의 전이 확률들로 부터 계산해 낼 수 있다.

n 차 마르코프 가정은 통계적 방법의 후처리에서 사용할 수 있는 문맥의 길이를 n으로 결정하므로, 후처리 성능은 마르코프 가정의 차수 n에 따라서 증가한다. 영문의 경우에 비터비 알고리즘으로 오류 교정 성능을 평가 실험하였을때, 1 차 전이 확률을 사용하였을 경우 약 16%의 교정율을 보이고, 2 차 전이 확률을 사용하였을 경우 약 35%의 교정율을 보인다는 보고가 있다[HULL 82].

〈표 2.1〉에서 Z를 정의하였던 대로 Context 부분과 Correction부분으로 Z를 나누어 표현하면 $t=2n+m$이 성립한다. 그러면 $\ln P(Z)$를 (7)식과 같이 $A(m,n)$과 $B(n)$의 합으로 표현할 수 있다.

$$\begin{aligned} & t = 2n+m \\ & \ln P(Z) = A(m,n) + B(n) \\ & A(m,n) = \ln P(Z_{n+1}|Z_1\cdots Z_n) + \cdots + \ln P(Z_{2n+m}|Z_{n+m}\cdots Z_{2n+m-1}) \\ & \qquad = \sum_{i=n+1}^{2n+m} \ln P(Z_i|Z_{i-n}\cdots Z_{i-1}) \\ & B(n) = \ln P(Z_1) + \ln P(Z_2|Z_1) + \cdots + \ln P(Z_n|Z_1\cdots Z_{n-1}) \end{aligned} \quad (7)$$

$B(n)$은 오류가 아닌 n개의 문자에 대한 확률이므로, $P(Z)$가 최대값이 되게 하는 Z를 찾는 경우에 영향을 미치지 못 한다. 따라서 $A(m,n)$이 최대값이 되게 하는 Z가 $P(Z)$도 최대값이 되도록 한다.

따라서 (3)식의 $G(X,Z)$를 조건부 독립과 n 차 마르코프 가정을 통해서 전개하고, (7)식의 $B(n)$항을 제거하면 (8)식의 $C(X,Z)$을 얻을 수 있다. (8)식의 $C(X,Z)$를 최대화하는 Z를 찾으면, 주어진 X를 Z로 교정할 수 있다.

$$C(X,Z) = \sum_{i=n+1}^{2n+m} \ln P(X_i|Z_i) + \ln P(Z_i|Z_{i-n}\cdots Z_{i-1}) \quad (8)$$

비터비 알고리즘

$C(X,Z)$가 최대값을 갖는 Z를 찾는 방법인 비터비 알고리즘(VA[주2])을 살펴본다.

VA은 1967년에 Viterbi에 의해 제안되었으며, 마르코프 과정에서 사후 확률(*posteriori* probability)을 최대화하는 상태전이를 찾는 문제(MAP[주3])에 대한 해결방법으로 알려져 있다[FORN 73].

m 개의 오류를 포함하는 문자열 X를 같은 길이의 Z로 교정하려고 할 때에 가능한 Z의 경우의 수는 r^m이다. 따라서 모든 경우 중에서 사후 확률이 가장 큰 Z를 선택함으로써 MAP를 해결할 수 있는데, r^m번의 비교연산이 필요하게 된다. 그러나 VA은 r^m번만큼 비교하는 것 대신에 $O(r^2)$의 계산량만으로 MAP를 해결할 수 있다.

MAP는 격자(*trellis*) 형식의 $26\times m$ 노드(node)로 이루어진 그래프(Graph)에서 최단 경로(Shortest Path)를 찾아내는 문제로 간주할 수 있다. 격자의 노드에는 혼동 확률값에 $-\ln P(X_i|Z_i)$ 값이 지정되어 있고, 격자의 에지(edge)에는 $-\ln P(Z_i|Z_{i-}$

[주2] Viterbi Algorithm
[주3] Maximum A posteriori probability Problem

$_{n}\cdots Z_{i-1}$)값이 지정되어 있다. 경로의 거리는 그 경로에 있는 에지와 노드의 값들을 합산한 값으로 (8)식의 $-C(X,Z)$로 정의한다.

2.2 중복 표현된 정보

후처리에서 사용할 수 있는 정보는 마르코프 가정의 차수에 의해서 결정되며, n차 전이 확률을 저장하는데 필요한 기억공간은 $O(r^n)$이다. 따라서 공간의 제약으로 인해서 높은 차수의 전이 확률을 후처리에 이용하기 어렵다. 통계적인 방법의 성능을 결정 짓는 가장 중요한 요소는 공간 효율이다. 이 절에서는 전이 확률에서 중복 표현된 정보를 찾아내고 제거하는 방법을 살펴 본다.

한글의 경우는 영문에 비하여 문자 집합의 크기 r이 크다. 영문의 문자 집합은 26개와 공란 1 개를 합한 r=27개인 반면에, 한글의 경우 비교적 문자 집합의 크기가 작은 N-바이트 코드의 문자 집합도 33개의 자소와 1개의 공란으로 r=34개이다. 그 외에 조합형의 경우 67개의 자소와 1개의 공란으로 68개이고, 완성형 코드의 경우는 2,350개의 음절과 1개의 공란으로 2,351개이다.

따라서 한글의 경우에는 전이 확률를 저장하는데 필요한 기억공간의 크기가 영문의 경우에 비해 더욱 급격하게 증가한다. 한글의 경우에 영문에 비하여 차수 n이 증가할 수록 더 많은 전이 확률이 중복되어서 표현된는 것을 알 수 있다. 한글의 경우에 중복 표현된 전이 확률을 제거하지 않으면 현실적인 기억공간에서 만족할만한 오류 교정 성능을 얻기 어렵다[황호정 94].

전이 확률에서 중복 표현된 정보는 무엇일까? 전이 확률의 중복 표현은 다른 차수의 전이 확률 사이에서 발생한다. 그 중복 표현되는 경우는 전이 확률의 값이 의미 없는 경우와 낮은 차수의 전이 확률을 통해서 현재 차수의 전이 확률값을 알 수 있을 경우이다.

전이 확률의 값이 의미 없는 경우

n차 전이 확률 $P(Z_m|Z_{m-n}\cdots Z_{m-1})$의 값이 무의미한 경우는 $P(Z_{m-n}\cdots Z_{m-1})=0$ 일 경우이다. 왜냐하면 n차 전이 확률은 n개의 문자가 나타난 경우에 Z_m이 나타날 확률이므로, 그 확률이 0이면 일어나지 않는 상태를 가정한 전이 확률이 되기 때문이다. 이 경우는 (9)식에서 보듯이 낮은 차수의 전이 확률값이 0인지를 통해서 알 수 있다.

$$
\begin{aligned}
&P(Z_{i-1}|Z_{i-n}\cdots Z_{i-2})=0\ ,\ n+1\le i\le 2n+m\\
&\Rightarrow P(Z_{i-n}\cdots Z_{i-2}Z_{i-1})=0\\
&\Rightarrow P(Z_{i-n}\cdots Z_{i-2}Z_{i-1}Z_i)=0\\
&\Rightarrow P(Z_i|Z_{i-n}\cdots Z_{i-1}) \text{ is not valid for all } Z_i\in S
\end{aligned}
\qquad (9)
$$

오류 교정 과정에서 이 경우의 상태 전이는 일어 날 수 없으므로, 전이 확률에 대한 기억공간을 할당하지 않음으로써 정보의 손실 없이 공간 효율을 높일 수 있다. 그 방법으로는 계층적 구조로 표현하는 기법과 마지널 인덱싱 기법(MIM[주4])이 있다.

낮은 차수의 전이 확률로 부터 알수 있는 경우

n-1 차 전이 확률이 0인 상태 전이를 포함하는 어떤 n차 전이 확률도 0 이므로 (10)식이 성립한다.

$$
\begin{aligned}
&P(Z_i|Z_{i-(n-1)}Z_{i-(n-2)}\cdots Z_{i-1})=0\ ,\ n+1\le i\le 2n+m\\
&\Rightarrow P(Z_{i-(n-1)}\cdots Z_i)=0\\
&\Rightarrow P(Z_{i-n}Z_{i-(n-1)}\cdots Z_i)=0\\
&\Rightarrow P(Z_i|Z_{i-n}\cdots Z_{i-1})\ ,\text{for all } Z_{i-n}\in S
\end{aligned}
\qquad (10)
$$

(10)식을 통해서 n-1 차 전이 확률값이 0임을 통해서 n차 전이 확률의 값을 알아낼 수 있으므로, 이 경우에 속하는 n차 전이 확률을 제거함으로써 공간 효율을 높일 수 있다. 이 경우의 중복 표현된 정보를 제거할 수 있도록 MIM을 개선하였다.

2.3 확률값의 표현

2.2에서는 구조적인 면에서 전이 확률이 중복 표현되는 것을 살펴보았다. 이 장에서는 확률값 자체를 적은 공간에 정보의 손실을 줄이면서 저장할 수 있는 방법을 살펴본다.

이진 n-gram과 전이 확률

전이 확률은 실수값이기 때문에 일반적으로 전이 확률 하나를 저장하는데에는 4바이트(byte)가 필요하다. 극단적으로 확률값을 정량화하여(quantize) 표현하는 방법으로 이진(Binary) n-gram 방법이 있다. 이 방법은 확률값을 0과 1로 정량화하여 표현하므로 상태 전이 하나를 저장하는데, 1 bit만을 사용한다. 전이 확률을 실수값으로 저장하는 방법에 비하여 32분의 1의 공간만을 사용한다. 이 방법은 오류 교정의 경우에는 너무 많은 교정 후보를 제시하고, 그 후보를 평가할 수 있는 정보

[주4] Marginal Indexing Method

가 없기 때문에 교정성능은 나쁘지만, 오류 탐지에서는 크게 성능이 저하되지는 않는다. [HULL 82]

전이 확률값을 표현하는 방법으로 불리안(Boolean)값과 실수(Real)값으로 표현하는 2가지 극단적인 방법이 있는데, 그 중간의 절충적인 형태를 생각해볼 수 있다. 그 방법은 정수(integer)값으로 전이 확률을 표현하는 방법으로 이진 n-gram 방법의 공간 효율과 실수값으로 표현하는 방법의 교정률을 동시에 얻을 수 있다.

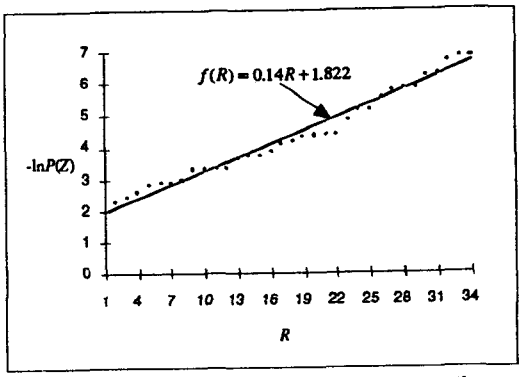
실수값을 정수값으로 정량화하여 4분의 3의 데이타를 버림에도 불구하고, 거의 차이없는 교정율을 제공할 수 있는 것은 전이 확률과 그 순위 사이의 상관관계를 이용하기 때문이다. 전이 확률값 대신에 전이 확률의 순위를 저장하고, 그 순위로 부터 전이 확률값을 최소자승법으로 추정해 냄으로써 비터비 알고리즘을 그대로 적용할 수 있도록 개선한다. 그 결과로 순위에서 전이 확률값에 대한 근사값을 무시할만한 오차로 추정할 수 있으므로, 거의 실수값으로 전이 확률을 저장하는 것과 다름 없는 오류 교정률을 제공한다.

출현 빈도와 빈도 순위간의 상관 곡선

중국 문자의 경우에 빈도의 순위와 출현빈도의 사이에서는 지수 함수(exponential function)적인 관계가 있다는 보고가 있다[WU 94]. 한글의 경우도 예외는 아니어서, 빈도의 순위와 출현 빈도 사이에서는 지수함수적인 상관관계가 있다. 따라서 오류 교정에서 사용하는 값인 $\ln P(Z)$은 그 순위와 비례하는 상관 관계가 있으므로, $\ln P(Z)$을 빈도 순위 R[95]로 부터 1 차 다항식(polynomial)으로 근사(approximate)할 수 있다. 다항식의 매개변수 a와 b는 최소자승법(LSM[96])을 사용하여 (11)식과 같이 산정할 수 있다.

$$-\ln P(S_R) \approx f(R)$$
$$f(R) = aR + b$$
$$a = \frac{6(r+1)\sum_{R=1}^{r}\ln P(S_R) - 12\sum_{R=1}^{r}(R\cdot\ln P(S_R))}{r^2-1}$$
$$b = \frac{6\sum_{R=1}^{r}(R\cdot\ln P(S_R)) - 2(2r+1)\sum_{R=1}^{r}\ln P(S_R)}{r(r-1)}$$
$$E = \sum_{R=1}^{r}(f(R)+\ln P(S_R))^2 : \text{Square Error} \qquad (11)$$

한편 2 차 이상의 다차 다항식으로 근사하면 오차(E)는 줄어들지만 오차의 감소폭에 비하여 계산량이 많이 증가하고, 1 차 다항식으로 근사하여도 오차를 무시할만함으로, 1 차 다항식으로 근사하는 방법을 본 논문에서는 선택했다. 〈그림 2.2〉는 N-바이트 코드의 자소집합[97]에 대한, 0 차 전이 확률 $\ln P(Z)$과 R 사이의 상관 관계 곡선이며, $f(R)$은 LSM을 사용하여 구한 $-\ln P(Z)$을 근사하는 직선 방정식이다.



〈그림 2.2〉 $-\ln P(Z)$와 빈도 순위(R) 사이의 상관 곡선
(a=0.14, b=1.822, E=1.4)

VA에서 사용되는 $-\ln P(Z)$를 실수값으로 저장하지 않고, 빈도 순위를 정수값으로 저장하여 $aR+b$값으로 대신하므로, 한번의 곱하기 연산으로 약 전체 사용 기억 공간의 4분의 3을 절약하는 효과가 있다. 비록 1 차 다항식으로 $-\ln P(Z)$를 근사하므로 오차(E)는 발생하지만 〈그림 2.2〉에서와 같이 오차는 무시할만하다.

제 3 장 설계 및 실험

이 장에서는 2 장에서 소개한 방법으로 전이 확률을 저장할 수 있는 프로그램을 설계하고,

[95] Rank
[96] Least Square Method
[97] r=34

각 구조에서 전이 확률에 접근하는 알고리즘을 설계한다. 한정된 기억공간 안에 각 방법으로 가능한 높은 차수의 전이 확률을 저장할 수 있도록 학습 텍스트를 반복해서 읽으면서 설정된 기억공간이 가득 찰 때까지 전이 확률의 차수를 높여가는 방법으로 전이 확률을 산출한다.

학습 텍스트를 입력으로 받아서, n 차의 전이 확률을 각각의 구조에 저장하여 출력한다. n 차 전이 확률은 학습 텍스트의 n+1 gram[주8] 의 출현 빈도로 부터 (12)식에 의해서 산출한다.

$$P(Z_{n+1}|Z_n\cdots Z_2Z_1) = \frac{P(Z_{n+1}Z_n\cdots Z_2Z_1)}{P(Z_nZ_{n-1}\cdots Z_2Z_1)} = \frac{freq(Z_{n+1}Z_nZ_{n-1}\cdots Z_2Z_1)}{freq(Z_nZ_{n-1}\cdots Z_2Z_1)} \quad (12)$$

3.1 모든 전이확률을 무조건 저장하는 기법

일반적으로 사용하는 방법인 모든 전이 확률을 무조건 저장하는 방법(Exhaustive Method)은 가장 많은 기억 공간을 사용하며, 접근 속도가 가장 빠른 방법이다. 이 방법으로 통계적인 방법이 주로 실험 되었기 때문에 통계적인 방법은 교정 속도는 빠르지만 교정율을 낮은 것으로 알려져 있다[민병우 91]. 이 방법을 소개하는 것은 이 방법을 기준점으로 삼아 개선된 다른 방법들을 비교를 하기 위해서이다.

1 차부터 n차까지의 전이 확률에 -ln을 취한 값을 저장할 수 있어야 한다. 값의 범위는 $0 \le -\ln P(Z_n|Z_1..Z_{n-1}) \le \infty$ 이고, 실수값으로 저장한다. 실수값을 표현하는데 4바이트가 소요된다고 가정한다. 무한대값은 범위에 속하지 않는 음수값으로 지정해서 사용한다. 0 차부터 n 차까지 전이 확률의 가지수는 (13)식과 같다.

$$\sum_{i=0}^{n+i} r^i = \frac{r^{n+2}-1}{r-1} \quad (13)$$

따라서 모든 전이 확률을 저장하는 방법을 사용하였을 경우에 (13)식 가짓수의 4배의 기억공간이 요구된다. 한글을 표현할 수 있는 코드 체계들 중에서 완성형 코드 체계, 조합형 코드 체계, N-바이트 코드 체계에서 모든 전이 확률을 저장하기 위해서 필요한 기억공간은 〈표 3.1〉과 같다.

〈표 3.1〉 모든 전이 확률을 저장하기 위해서 필요한 기억공간

심볼 집합의 종류	크기	1 차	2 차	3 차	4 차
영문 코드	27	3.0 KB	79.8 KB	2.1 MB	56.8 MB
N-바이트 코드	34	4.7 KB	158.2 KB	5.3 MB	178.6 MB
조합형 코드	68	18.3 KB	1.2 MB	82.8 MB	5,629.1 MB
완성형 코드	2,351	21.1 MB	48.4 GB	111.2 TB	261,402 TB

3.2 계층적인 구조로 표현하는 기법

(9)식에서 정의한 의미없는 전이 확률과 (10)식에서 정의한 예상 가능한 전이확률을 제거하기 위한 방법으로 전이 확률을 계층적인 구조로 구현하는 기법이다. 그 방법은 먼저 2-gram을 산출한 후 2-gram에서 발생한 상태 전이만으로 다시 심볼 집합을 구성한다. 그리고 그 심볼 집합의 원소로 이루어진 2-gram을 구성하면, 4-gram 테이블을 얻을 수 있다. 따라서 2-gram에서 발생하지 않은 상태 전이는 4-gram테이블을 구성하면서 제거할 수 있다.

이 방법은 2-gram정도만을 구하는 경우에는 빠른 접근 속도와 좋은 공간 효율을 제공하지만[박혜준 94], 2-gram에서 4-gram으로, 4-gram에서 8-gram을 산출하기 때문에 4-gram이상을 구하기에는 적합하지 않다. 왜냐하면, 4-gram을 구할 경우에 3-gram에서 발생하지 않는 상태전이에 공간이 할당되므로 필요한 기억공간이 급증하기 때문이다.

완성형 코드의 문자집합을 사용하여 2-gram 테이블을 구축하는 경우, 모든 전이확률을 저장하면 약 21.5MB가 요구되는 반면에, 계층적 구조로 표현하였을 경우에는 4.7MB에 표현할 수 있다. 이 방법은 단순하여 계산량이 적고, 2-gram을 구현하기에는 적합하다.

3.3 마지널 인덱싱 기법

마지널 인덱싱 기법(MIM[주9])이란, 트리(Tree) 구조로 전이 확률을 표현하는 방법으로 트리의 각 레벨은 전이 차수에 해당한다. 트리 구조이므로 각 레벨에서 출현하지 않은 심볼들에 대한 자식 노드를 갖지 않으므로, (9)식에서 정의한 무의미한 전이 확률을 제거할 수 있다.

이 방법은 계층적 구조로 표현하는 것에 비하여 순차적으로 차수가 증가하면서 낮은 차수에서 발생하지 않는 상태전이를 높은 차수의 상태전이에 공간을 할당하지 않으므로, (9)식에서 정의한 중복 표현된 정보를 완전하게 제거할 수 있다.

[주8] $Z_{n+1}Z_nZ_{n-1}\cdots Z_2Z_1$

[주9] Marginal Indexing Method

그러나 트리 구조로 표현할 때 링크 필드가 부가적으로 필요하게 된다. 부가적으로 필요한 링크 필드를 제거하기 위해서 마지널 인덱싱을 선형화하는 방법이 제안되었다[HULL 82].

마지널 인덱싱을 선형화(Linearization)하는 기법은 링크 필드를 제거하고, 트리의 각 레벨마다 하나의 어레이를 선언하여 전이확률을 저장한다. 이 때에 다음 레벨을 접근할 때에서는 부모 레벨에 있는 전이 확률값이 0 이 아닌 노드의 갯수를 셈으로써 다음 레벨의 노드의 인덱스(index)를 계산하여 접근한다. 따라서 마지널 인덱싱에 비해서 접근 속도는 느려지지만 링크 필드를 제거할 수 있게 된다. 영문의 경우 3-gram을 선형화된 마지널 인덱싱 기법으로 저장하였을 경우 약 70%의 공간을 절약할 수 있었다고 한다[HULL 82].

3.4 개선된 마지널 인덱싱 기법

3.3 절에서 마지널 인덱싱 기법으로 (9)식에서 정의된 무의미한 전이확률을 완전히 제거할 수 있음을 살펴 보았다. 그러나 (10)식에서 정의한 예상 가능한 전이 확률은 제거되지 못 한다. 따라서 (10)식의 전이 확률의 중복 표현을 제거할 수 있도록 마지널 인덱싱을 개선할 필요가 있다. (10)식에서 전이 확률을 제거할 수 있는 단서는 $P(Z_i|Z_{i-(n-1)}\cdots Z_{i-1})=0$ 이다. 이 조건을 통해서 알 수 있는 것은 마지널 인덱싱의 n레벨에서 나타날 수 있는 상태 전이의 갯수이며, 자식 노드의 크기가 된다.

노드의 크기를 (10)식의 조건을 만족하지 않는 상태 전이의 갯수로 줄임으로써 (10)식의 중복 표현된 확률을 제거할 수 있다. 즉 기존의 마지널 인덱싱 기법은 노드의 크기는 r로서 일정한 반면에 개선된 마지널 인덱싱은 부모 노드들의 값에 의해서 노드의 크기가 변화된다는 점이 다르다.

노드의 크기를 결정하기 위해서는 부모 노드들 중에서 $P(Z_i|Z_{i-(n-1)}\cdots Z_{i-1})\neq 0$인 경우의 수를 알아내어야 한다. 이 작업은 최대 r개의 전이 확률값을 참조해야 한다. 그리고, n차 전이 확률값을 참조하기 위해서는 n-1차전이 확률을 참조해서 n 그 노드의 크기를 알아내어야 한다. 따라서 재귀적인 방법으로 그 전이 확률값을 참조하게 된다. 그러므로 전이 확률을 하나 참조하는데에 필요한 연산이 복잡하다.

따라서 이방법은 (9)식과 (10)에서 정의한 중복 표현된 전이 확률을 완전히 제거할 수 있는 방법이지만, 검색시간이 많이 걸리는 문제점이 있다. 결과적으로 이방법의 검색 속도를 향상시키기 위해서 검색을 위한 정보를 추가적으로 유지할 필요가 있다.

3.5 순위에서 확률을 추정하는 기법

순위에서 확률을 추정하는 방법은 구조적으로 중복성을 제거하는 것이 아니므로, 앞에서 살펴 보았던 모든 방법과 병행해서 사용할 수 있다.

그중에서 3.1절의 모든 전이 확률을 저장하는 방법에 적용하면, n차 전이 확률이 구해져 있다고 가정한다. 각 상태에서의 전이 확률을 정렬해서 확률이 높은 것부터 낮은 순서로 번호를 붙임으로써 R을 얻어낸다. 그리고 (11)식에 의해서 a와 b를 산출한다. 실수값으로 저장되어 있던 전이 확률을 버리고, 순위 정보와 각 상태마다 a와 b를 가지고 대신한다.

3.1절의 방법에 적용을 하면, (14)식에서와 같은 양의 공간을 제거하는 효과가 있다.

$$
\begin{aligned}
&Mem\_Use_{3.1} - Mem\_Use_{3.5} \\
&= 4\sum_{i=0}^{n+1} r^i - (\sum_{i=0}^{n+1} r^i + 8\sum_{i=0}^{n} r^i) \\
&= 3\frac{r^{n+2}-1}{r-1} - 8\frac{r^{n+1}-1}{r-1} \\
&= \frac{r^{n+1}(3r-8)+5}{r-1}
\end{aligned}
\qquad (14)
$$

(14)식에서 보듯이 이 방법으로 얻을 수 있는 공간 효율은 심볼 집합(r)의 크기가 큰 경우에 더 높아진다. 따라서 한글과 같이 영문에 비해 심볼 집합의 크기가 큰 경우에 더 효과를 발휘할 수 있다.

제 4 장 결론

통계적 방법을 사용한 후처리의 성능은 공간 효율에 달려 있다는 가정에서 공간 효율을 높이기 위해 의미 없는 전이 확률, 예상 가능한 전이 확률을 정의하여 제거할 수 있는 방법을 제안하였다. 공간 효율을 높일 수록 접근 속도가 느려지는 현상이 일어났다. 가장 공간 효율이 높은 방법인 선형화된 마지널 인덱싱 기법의 개선 기법은 전이 확률의 갯수에 검색속도가 선형적으로 증가하는 현상을 보였다.

순위 정보를 이용하여 전이 확률을 근사할 경우는 심볼집합의 크기가 클 수록 많은 공간을 제가함을 알 수 있었다.

검색 속도를 향상시킬 수 있는 정보가 필요하므로, 확률값 자체의 공간 효율을 높이기 위해서 사용했던 순위 정보를 다시 검색에 이용함으로써 중복된 정보를 극소화하면서 검색속도를 향상시킬 수 있는 방법을 고려 중이다.

참고 문헌

ABE 71 Abe, K. & Fukumura, T. "Word-seraching methods in character recognition system with dictionary," *Systems, Computers, Controls, 2(5)*, 1971, pp. 1~9..

BLED 66 Bledsoe, W. W. & Browning, J. "Pattern recognition and reading by machine.," *Pattern Recognition*, 1966, pp. 301~316.

FORN 73 Forney, G. D. "The Viterbi Algorithm," *Proceeding Of The IEEE Vol. 61, No. 3, March*, 1973, pp. 268~277.

HALL 80 Hall, P. A. V., & Dowling G. R. "Approximate String Matching," *ACM Computing Surveys*, 1980.12, pp. 381~402.

HART 83 Hartigan, J. A. "Bayes Theory," *Springer-Verlag*, 1983.

HULL 82 Hull, J. J., & Srihari, S. N. "Experiments in Text Recognition with Binary n-Gram and Viterbi Algorithms," *IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. PAMI-4, No. 5, September*, 1982, pp. 520~530.

KONN 93 Konno, A., Hongo, Y. "Postprocessing Algorithm based on the Probabilistic and Semantic Method for Japanese," *IEEE*, 1993, pp. 646~649.

LEE 93b Lee, S. W., Kim, E. S., & Min, B. W. "Efficient Postprocessing Algorithms for Error Correction in Handwritten Hangul Address and Human Recognition," *IEEE*, 1993, pp. 232~235.

MASE 80 Masek, W. J., & Paterson M. S. "A Faster Algorithm Computing String Edit Distances," *Journal Of Computer And Systems Sciences*, 1980.2, pp. 18~31.

OKUD 76 Okuda, T., Tanaka, E., & Kasai, T. "A Method for the Correction of Garbled Words Based on the Levenshtein Metric," *IEEE Transactions On Computers*, 1976, pp. 172~178.

OOMM 87 Oommen, B. J. "Recognition of Noisy Subsequences Using Constrained Edit Distances," *IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. PAMI-9, No. 5, September*, 1987, pp. 676~685.

RAVI 67 Raviv, J. "Decision Making in Markov Chains Applied to the Problem of Pattern Recognition," *IEEE Transactions On Information Theory Vol It-3 No. 4*, 1967, pp. 536~551.

REVU 84 Revuz, D. "Markov Chains," *Elsevier Science Publishers B.V.*, 1984.

RISE 74 Riseman, E. M., & Hanson, A. R. "A Contextual Postprocessing System for Error Correction Using Binary n-Grams," *IEEE Transactions On Computers*, 1974.5, pp. 480~493.

SHIN 79 Shinghal, R. "A bottom-up and top-down approach to using context in text recognition," *International Journal Of Man-Machine Studies*, 1979, pp. 201~212.

SRIH 83 Srihari S. N., Hull, J. J., & Choudhari, R. "Integrating Diverse Knowledge Sources in Text Recogntion," *ACM Transactions On Office Information Systems*, 1983.1, pp. 68~87.

SRIH 85 Srihari, S. N. "Computer Text Recognition and Error Correction," *IEEE Computer Society*, 1985, pp. 1~3.

TANA 87 Tanaka, E. & Kojima, Y. "A High Speed String Correction Method Using a Hierarchical File," *IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. PAMI-9, No. 6, November*, 1987, pp. 806~815.

THOM 91 Thomas, K. "Entropy," *Dr. Dobb's Journal*, 1991.2, pp. 32~34.

TOUS 78 Toussaint, G. T.,& Shinghal, R. "Experiments in Text Recognition with the Modified Viterbi Algorithm," *IEEE Transactions In Pattern Analysis And Machine Intelligence, Vol. PAMI-1*, 1978, pp. 184~193.

ULLM 77 Ullmann, J. R. "A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words," *Computer Journal*, 1977, pp. 141~147.

WAGN 74 Wagner, R. A., & Fischer, M. J. "The String-to-String Correction Problem," *Journal Of The ACM*, 1974, pp. 168~173.

WU 94 Wu, R. Y. & Tsai, W. H. "Frequency-Rank Curves and Entropy for Chinese Characters and Words," *Chinese Langauge Computer Society*, 1994, pp. 37~52.

강재우 89 강재우 "접속정보를 이용한 한글 철자 및 띄어쓰기 검사기의 설계 및 구현," *한글 및 한국어 정보처리 학술 발표논문집*, 1989, pp. 3~9.

김병희 93 김병희, 송만석 "형태소 접속 특성과 인접 말마디 정보를 이용한 형태소 분석기," 연세대학교 석사학위 논문, 1993.

민병우 91 민병우, 이성환 "문자 인식을 위한 오인식 수정 기술," *정보과학회 Vol. 9. No. 1*, 1991.2, pp. 7~13.

박종만 90 박종만, 김영찬 "한글 맞춤법 검사 시스템의 개발," *제2회 한글 및 한국어정보처리 학술발표 논문집*, 1990, pp. 11~15.

박영환 91 박영환, 송만석 "말뭉치에 기반한 형태소 분석기 및 철자 검사기의 구현," *연세대학교 석사학위 논문*, 1991, pp. 1~4.

박혜준 94 박혜준, 송만석 "말뭉치에서의 품사 꼬리달기 시스템의 구현," *연세대학교 석사학위 논문*, 1994, pp. 19~28.

이영식 93 이영식, 채영숙, 윤애선, 권혁철 "한국어 철자 오류 교정 시스템," *정보과학회 인공지능연구회 춘계 학술발표회 논문집*, 1993, pp. 25~38.

이원일 93 이원일, 홍남희, 이종혁, 이근배 "Binary N-gram과 형태소 분석기를 이용한 한국어 철자 교정기," *한국정보과학회 93년 봄 학술발표논문집*, 1993, pp. 813~816.

채영숙 92 채영숙, 이영식, 권혁철 "한글 철자 오류 교정 시스템," *1992년도 제 4회 한글 및 한국어 정보처리 학술발표 논문집*, 1992, pp. 459~468.

황호정 94 황호정, 도정인, 권혁철 "한글 문자 인식을 위한 후처리기의 개발과 속도 개선," *제2회 문자 인식 워크샵*, 1994, pp. 180~188.