띄어쓰기 및 철자 오류 동시교정을 위한 통계적 모델

(A Joint Statistical Model for Word Spacing and Spelling Error Correction Simultaneously)

노 형 종 † 차 정 원 †† 이 근 배 †††
(Hyungjong Noh) (Jeongwon Cha) (Gary Geunbae Lee)

요 약 본 논문에서는 띄어쓰기 오류와 철자 오류를 동시에 교정 가능한 전처리기를 제안한다. 제시된 알고리즘은 기존의 전처리기 알고리즘이 각 오류를 따로 해결하는 데에서 오는 한계를 극복하고, 기존의 noisy-channel model을 확장하여 대화체의 띄어쓰기 오류와 철자 오류를 동시에 효과적으로 교정할 수 있다. N-gram과 자소변환확률 등의 통계적 방법과 어절변환패턴 사전을 이용하여 최대한 사전을 적게 이용하면서도 효과적으로 교정 후보들을 생성할 수 있다. 실험을 통해 현재 단계에서는 만족할 만한 성능을 얻지는 못하였지만 오류 분석을 통하여 이와 같은 방법론이 실제로 효용성이 있음을 알 수 있었고 앞으로 더 많은 개선을 통해 일상적인 대화체 문장에 대해서 효과적인 전처리기로서 기능할 수 있을 것으로 기대된다.

키워드 : 띄어쓰기 교정, 철자 교정, 텍스트 전처리

Abstract In this paper, we present a preprocessor which corrects word spacing errors and spelling correction errors simultaneously. The proposed expands noisy-channel model so that it corrects both errors in colloquial style sentences effectively, while preprocessing algorithms have limitations because they correct each error separately. Using Eojeol transition pattern dictionary and statistical data such as n-gram and Jaso transition probabilities, it minimizes the usage of dictionaries and produces the corrected candidates effectively. In experiments we did not get satisfactory results at current stage, we noticed that the proposed methodology has the utility by analyzing the errors. So we expect that the preprocessor will function as an effective error corrector for general colloquial style sentence by doing more improvements.

Key words : word spacing, spelling error correction, text normalizer

1. 서 론

메신저나 SMS등을 통한 대화시스템이 늘어나면서 구어체 문장의 텍스트 전처리를 효율적으로 해야 할 필요성 또한 커지고 있다. 일반적인 문어체에 대해서도 그렇지만, 구어체 문장을 전처리하는 과정에 있어서 띄어쓰기와 철자 오류를 교정하는 것은 가장 기본적인 문제들 중 하나이고 이를 해결하기 위해 여러 가지 방법들이 제시되어 왔다.

띄어쓰기 교정을 해결하기 위해 많이 이용된 방법은 크게 통계적 방법과 규칙을 이용한 방법으로 나눌 수 있는데 통계적 방법은 주로 어절이나 음절 간 n-gram을 이용하거나[1,8], noisy-channel model을 이용하는 [2] 방식이었다. 규칙을 이용하는 방법은 언어학적인 지식을 이용한 휴리스틱 알고리즘을 채택하여 그 언어 특성을 반영하여 띄어쓰기 문제를 해결하고 있었다[3]. 띄어쓰기 문제는 주로 공백문자 자체를 사용하지 않는 일본어와 중국어, 조사나 어미의 변화가 심한 한국어에 대해 다루어져 왔다.

철자 교정을 위해 제안되었던 방법들은 문장 내의 각 단어에 대해 이 단어가 단어 사전 내에 포함되어 있지

· 본 연구는 ETRI 연구 "음성대화 시스템을 위한 효율적인 후처리 방법" 과제의 지원을 받아 수행되었음
† 비 회 원 : 포항공과대학교 컴퓨터공학과
nohhj@postech.ac.kr
†† 종신회원 : 창원대학교 컴퓨터공학과 교수
jcha@changwon.ac.kr
††† 종신회원 : 포항공과대학교 컴퓨터공학과 교수
gblee@postech.ac.kr
논문접수 : 2006년 8월 13일
심사완료 : 2006년 8월 31일

않다면 그 단어를 철자 오류가 포함된 단어로 보고, 이를 대체할 만한 후보를 역시 단어 사전에서 찾아 그 후보 중 가장 대체 확률이 높은 단어로 대체하는 과정을 따르고 있다. 그 후보를 찾는 과정에서 단어 사이의 n-gram 등의 문맥 정보를 이용하거나, 철자 오류가 포함된 단어와 그 단어의 후보 간의 근접도를 측정하기 위해 edit-distance 등의 기준을 이용하기도 한다[4,5]. 그런데 이와 같은 접근법은 하나의 어절이 하나의 단어와 일대일 대응이 될 때에 가능한 방법들이다. 그리고 단어 사전의 크기가 모든 어휘를 포함할 만큼 커야 할 뿐만 아니라 신조어나 축약어 등등 대화체 문장에서 자주 나타나는 단어들에 대해서는 이들을 모두 교정 대상 단어로 분류하게 되어 전혀 엉뚱한 후보들을 생성시켜 성능이 낮을 수밖에 없다. 이 방법이 효과적으로 적용되는 경우는 어절과 단어의 단위가 일치하는 영어 등의 언어에서 축약어나 신조어가 적은, 즉 교정 대상 단어에서 교정 결과 단어를 문자간의 근접도만을 이용해 추정이 가능한 문어체 문장을 다룰 때이다. Edit-distance등의 문자간의 근접도를 이용하는 방법의 한계를 극복하기 위해 제안된 방법 중 하나는 교정 대상 단어와 교정 결과 단어의 앞뒤 문맥을 이용하여 그 유사도가 높을수록 그 단어로 교정될 가능성이 높도록 하는 것이었다[6]. 이 방법을 통해 전혀 비슷하지 않은 문자로의 전환도 후보로 생성해 낼 수 있다. 그런데 이 방법은 각각의 단어가 어떤 단어들과 쓰이는지 자료를 모을 수 있는 인터넷 포털사이트의 검색단어 정보 등의 방대한 로그가 필수적으로 필요하기 때문에 일반적인 텍스트 전처리기의 방법으로는 쓰일 수 없다. 또 다른 방법으로는 영어권 언어와 달리 단어와 어절의 일대일 대응이 어려운 언어의 특색 때문에 일본어를 위해 제안된 모델도 있다[7]. 이 모델은 OCR로 인식된 문자들의 철자 오류를 교정하기 위해 제안된 것인데 각 문자의 변환 확률과 단어의 POS(part of speech) 태그간의 n-gram을 이용하고 있다. 우리가 제안하는 방식과 유사한 부분이 많은데 이 역시 단어 사전을 이용하는 것이 불가피하다. 그리고 띄어쓰기가 없는 일본어의 특성상 문장 내의 가능한 단어를 찾아내기 위해 문장 내의 모든 문자조합을 검색하기 때문에 POS 태거가 아닌 단순 전처리기로 쓰기에는 시간에 대한 비용이 너무 크다.

그 동안 제안되었던 여러 가지 알고리즘들을 나열하였는데 각각의 방법의 문제점들도 있지만 무엇보다도 큰 문제는 앞서의 방법들은 띄어쓰기 또는 철자 오류 둘 중 하나만을 독립적으로 다루고 있다는 것이다. 이 방법들은 두 가지 오류가 동시에 나타날 경우 성능이 떨어지거나 적용 자체가 힘들어지게 된다. 실제로 우리가 고쳐야 할 문장들은 띄어쓰기와 철자 오류 모두가 포함되어 있기 때문에 두 가지 오류를 동시에 효과적으로 교정하는 새로운 방법이 필요하다.

본 논문에서는 띄어쓰기와 철자 오류가 모두 포함된 입력 문장에 대해서 동시에 전처리가 가능한 새로운 통계적 방법을 제안한다. 이 방법은 기본적으로 noisy-channel model을 바탕으로 하고 있으며, 각 자소의 변환확률값과 어절변환패턴 사전을 이용하여 철자 교정 후보들을 생성한다. 생성된 후보들에 각각 띄어쓰기 교정을 위해 공백문자가 삽입된 후보가 더해지고, 이렇게 생성된 자소/어절 후보 경로에서 최종적으로 교정된 결과를 얻게 된다. 이 방법은 특히 자소변환확률을 이용하여 어절변환패턴 사전의 크기를 상당히 줄이고 사전 내에 있지 않은 패턴 또한 교정이 가능하도록 해준다.

본 논문의 내용은 다음과 같다. 2장에서는 통계적으로 자소 단위 오류 수정을 하는 이유와 데이타를 뽑아내는 과정을 서술한다. 3장에서는 통합 모델의 구체적인 정의와 알고리즘의 진행 과정을 설명한다. 4장에서는 일련의 과정을 실제로 구현한 뒤 실험을 통해 얻어낸 수치를 통해 여러 가지 분석을 실행한다. 마지막으로 본 논문의 결론과 앞으로 해야 할 연구들에 대해서 언급하도록 한다.

2. 자소 단위의 오류 수정

본 논문에서는 자소 단위의 변환확률값을 이용한 교정 방식을 이용하였다.

2.1 자소 단위 변환을 이용한 이유

철자 오류의 많은 부분은 자소의 변환만으로도 교정이 가능하다. 본 논문의 실험에 사용한 gnome irc[1] 말뭉치에서 태깅된 철자 오류 중 약 36%의 오류는 단순히 자소 변환만으로 교정이 가능한 형태였다. 자소 단위의 오류 수정 방법을 이용해 그만큼의 어절변환패턴 사전의 크기를 줄일 수 있게 된다. 또한 자소 단위의 수정 방법은 음절 단위에 비해 모델 자체의 크기가 작다. 많은 음절들의 변화를 자소 단위의 변환확률을 이용하면 적은 모델만으로도 초, 중, 종성의 조합을 이용해 교정이 가능하다. 또한 학습 말뭉치에서 나타나지 않았던 다양한 음절 변환도 자소 단위의 변환확률을 이용하여 교정이 가능하다는 이점이 있다. 단순히 음절 단위의 후보만을 말뭉치에서 뽑아낸다면 말뭉치에 나타나지 않은 음절의 교정은 불가능하게 된다. 이는 기존의 영어권 언어에서 많이 이용하는 하나의 교정 대상 단어를 다른 후보 단어로 치환하는 방법에서도 나타나는 문제점으로 이 문제를 해결하려면 학습을 위한 방대한 말뭉치가 요구된다. 하지만 자소단위의 후보 생성은 말뭉치에 나타나지 않은 음절이라 하더라도 자소조합을 통해 충분히

1) http://www.gnome.or.kr/irc/

후보를 생성해 낼 수 있고 많지 않은 말뭉치만으로도 신뢰성 있는 자소 단위 변환 정보를 얻을 수 있다. 다만, 수많은 자소조합이 가능하기 때문에 자칫 후보 과생성이라는 문제를 야기할 수 있다. 이를 해결하기 위해 우리는 일정 임계값보다 변환확률이 작은 자소들을 애초에 후보로 생성시키지 않거나, 또 생성된 후보가 음절로 조합될 때마다 가지치기를 하여 후보를 줄였다. 또한 한글에서 인정하지 않는 자소조합이 나타날 경우 후보에서 제외하는 방식으로써 후보 과생성 문제를 최대한 줄였다. 이렇게 교정대상 문장의 각 자소들로부터 변환 후보들을 생성하고 최적 경로를 찾는 방식으로 많은 철자 교정 문제를 해결할 수 있다.

2.2 자소 단위 변환 데이타 추출

우리는 띄어쓰기와 철자 오류가 포함된 채팅 말뭉치와 그 오류를 수작업으로 교정한 말뭉치 한 쌍을 문장간 평행하게 일대일 대응이 가능하도록 준비하였다. 그리고 각 어절들을 자소 단위로 비교를 하여, 철자 교정의 과정이 한 개의 자소 단위로 이루어지는 경우, 두 개의 자소 단위로 이루어지는 경우, 세 개 이상의 단위로 이루어지는 경우로 나누었다.

한 개의 자소 단위로 교정된 경우는 주로 발음을 실수로 혹은 고의적으로 다르게 표기한 것들이 많았다(같애요→같아요/ㅐ→ㅏ, 먹어여→먹어요/ㅕ→ㅛ). 또는 맞춤법 지식의 부재로 인한 경우도 있었다(희안한→희한한/ㅇ→ㅎ). 자소가 빠짐으로써 교정되는 경우도 자소 단위의 교정에 포함하였다(나와욧→나와요/ㅅ→X[2]). 이들로부터 각 오류 자소가 어떤 자소로 변환되면서 교정이 되는지 그 수를 세어서 자소 확률 변환 데이타로 구축하였다. 각각의 자소 변환 확률은 초, 중, 종성을 구분하여 계산하였다. 이는 성능의 향상과 함께 자소 후보의 과생성을 막고자 함이다.

두 개의 자소 단위 교정은 인접한 두 개의 자소가 한꺼번에 바뀌어 교정되는 패턴이다. 특히 음절 A와 B가 연속하여 나왔을 경우 A의 종성과 B의 초성이 연계되어 같이 변화하는 경우가 거의 대부분이었다. 이 역시 대부분이 발음을 맞춤법과 다르게 표기한 경우였는데 발음을 소리 나는 대로 쓰거나 고의적으로 바꾸어 쓰는 경우가 많았다(있자나요→있잖아요/Xㄴ→ㄴㅎㅇ, 촙오→초보/ㅂㅇ→Xㅂ). 이들은 두 개의 자소를 하나의 변환 단위로 보아 역시 각각의 단위변환 횟수를 통계 데이타로 이용하였다.

앞의 두 가지 변환을 자소 단위 교정으로 규정하고, 그 외의 교정 패턴은 자소 단위로 접근할 수 없기 때문에 음절들의 자체 변환으로 생각하였다. 오류 음절들의 변화가 너무 심해 단순한 자소 변환만으로 그 원래의 음절을 복원할 수 없거나 채팅시 많이 쓰이는 신조어나 은어, 혹은 초성만으로 이루어진 형태 등등이 이에 해당하였다(걍→그냥, 글쵸→그렇죠, ㄳ→감사). 이 패턴들은 음절들 혹은 어절 자체를 하나의 단위로 보아 어절변환 패턴 사전을 구성하였다. 자소 단위 외의 후보를 생성하는데 이 사전이 이용된다.

3. 띄어쓰기 및 철자 오류 교정을 위한 통합 모델

3.1 통합 모델의 필요성

전처리에 있어서 띄어쓰기 오류만을 교정하려고 한다면 이 문제는 각 음절 사이에 공백 문자를 삽입하느냐 마느냐의 문제로 정의될 수 있다. 만약 철자 오류만을 교정한다면 이는 문장 내의 각각의 어절 또는 음절의 오류 포함 유무 판단과 함께 교정 후보들을 생성하는 문제가 된다. 그런데 현재까지 띄어쓰기나 철자 교정에 효과적으로 알려진 알고리즘들은 이 오류들이 복합적으로 나타났을 경우에 제대로 동작하지 않는다.

그 이유는 서론에서 밝혔듯이 띄어쓰기에서 많이 쓰이는 방법은 음절이나 어절, 혹은 형태소 등의 단위에 대한 언어 모델(n-gram) 등을 이용하는 것인데 이는 그 단어나 문자들이 올바르다는 가정 하에서만 이루어질 수 있기 때문이다. 또한 문장 내에 철자 오류가 섞여 있다면 띄어쓰기 알고리즘이 적용하는 언어 모델은 철자의 오류를 고려하지 않고 적용되므로 올바로 이용할 수 없다.

철자 교정을 다루는 알고리즘은 주로 단어 사전과 문장 내에 나타나는 형태소를 edit-distance 등의 문자간의 근접성을 바탕으로 비교, 그 거리가 가까운 단어들을 후보로 생성하는 방식을 이용한다. 영어 등의 언어에서는 어절 자체가 하나의 단어가 되므로 바로 적용이 가능하지만 한글에서는 조사, 어미 등이 있으므로 이들을 통계적인 방법이나 사전을 이용하여 따로 처리하거나, 아니면 그 어절 전체를 사전과 비교할 수도 있다. 하지만 이 역시 주어진 문장의 띄어쓰기를 바탕으로 어절을 나눈 후에 알고리즘을 적용하게 되어, 만약 주어진 문장에 띄어쓰기 오류가 포함된다면 현재까지 알려진 철자 교정 알고리즘으로는 올바른 후보를 생성할 수가 없다. 어미, 조사 등은 특히 그 띄어쓰기된 정보를 이용하여 구분하는 방법을 쓰게 되는데 만약 그 띄어쓰기 정보가 잘못되어 있다면 이와 같은 분석은 오류를 발생시키는 원인이 된다. 따라서 실제로 철자오류와 띄어쓰기가 같이 나타나는 빈도수가 매우 높은 구어체 문장에서는 이 알고리즘들을 실용적으로 이용하기가 매우 어렵다. 만약

2) X는 자소가 비어있음을 표기

기존의 알고리즘들을 이용한다면 띄어쓰기를 한 후에 철자 교정을 하거나, 반대로 철자 교정을 먼저 한 후에 띄어쓰기를 하는 방식으로 이용할 수밖에 없는데 이와 같은 방법은 앞서 언급한 바와 같이 철자 교정이나 띄어쓰기를 하는 알고리즘이 일단 둘 중 하나의 오류는 없다는 가정 하에 이루어지기 때문에 완벽한 성능을 낼 수가 없다. 그래서 이를 통합하여 동시에 교정이 가능한 전처리기를 생각하였고, 결국은 이 두 가지 오류는 동시에 교정해야 한다는 결론을 내렸다. 이를 위해 우리는 새로운 통합 모델을 다음과 같이 제시한다. 이 모델은 띄어쓰기와 철자 교정을 따로 구분하지 않고 같이 처리함으로써 기존의 알고리즘을 이용하면서 생기는 문제를 해결하였다.

3.2 문제 정의

띄어쓰기 오류와 철자 오류가 모두 포함된 문장 T가 주어지면, 우리는 이 문장으로부터 교정 후보 C들을 생성해 내고, 그 중 최종적으로 교정될 확률이 가장 높은 새로운 문장 C'를 찾는다.

$$C' = \arg\max_C P(C \mid T). \quad (1)$$

3.3 모델 구성

주어진 문장 T는 기본적으로 음절 s_i와 음절 사이사이에 공백문자 b_i가 포함되어 있다.

$$T = s_1 b_1 s_2 b_2 s_3 b_3 \ldots s_n b_n, \text{ n은 음절의 개수.} \quad (2)$$

다시 음절 s_i는 초, 중, 종성 총 3 개의 자소로 구성된다. 구성 요소가 없는 자소는 'X'로 정의한다. b_i는 공백이 존재시 'B', 공백이 없을 시 'Φ'로 정의한다.

$$s_i = j_{i1} j_{i2} j_{i3} \ (j_{i1}\text{: 초성, } j_{i2}\text{: 중성, } j_{i3}\text{: 종성}). \quad (3)$$

C 역시 T와 같이 음절과 공백 문자로 정의할 수 있다.

$$C = s_1 b_1 s_2 b_2 s_3 b_3 \ldots s_n b_n. \quad (4)$$

이제 $C' = \arg\max_c P(C \mid T)$를 만족하는 C'를 구하기 위해 $P(C \mid T)$를 Bayes' Rule을 적용하여 풀면,

$$\begin{aligned} C' &= \arg\max_C P(C \mid T) \\ &= \arg\max_C P(T \mid C) P(C) / P(T) \\ &= \arg\max_C P(T \mid C) P(C). \end{aligned} \quad (5)$$

$P(C)$는 C를 이루는 각 음절(공백문자도 음절로 취급)의 trigram을 이용하여 구한다.

$$P(C) = \prod_{i=1}^{n} P(c_i \mid c_{i-1} c_{i-2}), \quad c = s \text{ 또는 } b. \quad (6)$$

그리고 $P(T \mid C)$는 각각의 자소의 변환확률값을 이용, 자소간 문맥을 고려하지 않고 이들의 곱으로 정의한다.

$$\begin{aligned} P(T \mid C) &= \prod_{i=1}^{n} P(s_i \mid s_i') \\ &= \prod_{i=1}^{n} [P(j_{i1} \mid j_{i1}') P(j_{i2} \mid j_{i2}') P(j_{i3} \mid j_{i3}')]. \end{aligned} \quad (7)$$

여기에 각각의 공백 문자 변환확률도 곱하면,

$$P(T \mid C) = \prod_{i=1}^{n} [P(j_{i1} \mid j_{i1}') P(j_{i2} \mid j_{i2}') P(j_{i3} \mid j_{i3}') P(b_i \mid b_i')]. \quad (8)$$

가 된다.

이로써 자소변환을 이용한 noisy-channel model이 정의되었다. 실제 구현시에는 $P(C \mid T)$에 로그를 취하여 누적로그값의 합들로 계산하였다. 자소간의 변환확률은 2.2에서 언급한 바와 같이 얻을 수 있고, 공백문자 변환확률 또한 말뭉치에서의 공백의 유무 전환 횟수를 세어 계산할 수 있다.

자소 변환 이외의 형태로 변화하는 철자 오류는 앞에서 언급했었던 두 개 이상의 자소변화패턴으로 처리한다. 이 역시 변환확률을 구해 놓았으므로 자소의 경우와 마찬가지 방법으로 후보를 생성할 수 있다. 후보가 생성되는 형태를 그림으로 나타내면 다음과 같다.

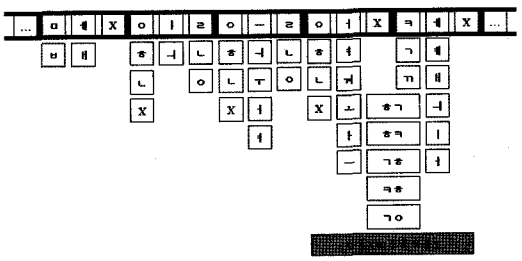

그림 1 자소단위 후보 생성의 예[3]. 입력 문장의 각 음절을 초, 중, 종성으로 분리하여 나열하였다. 각각의 자소에 대해서 그 자소가 다른 자소로 변환할 확률이 있는지를 자소변환패턴사전으로부터 탐색하여, 존재하는 변환 가능한 자소들을 모두 후보로 생성한다. 인접한 두 음절에서 앞 음절의 종성과 뒷 음절의 초성이 연계되어 변환되는 경우에 대해서도 자소 2개를 하나의 단위로 하여 후보가 생성된다. 어절 단위의 변환 후보는 어절변환패턴사전으로부터 탐색되어 후보로 생성된다.

그림 1이 나타내는 것은 문장을 자소로 분리한 후 각각의 후보를 생성시킨 모습의 일부분이다. 그림에서 맨 윗줄의 자소들이 입력 문장을 자소단위로 나누어 놓은

3) 예제문장은 "데체메일을어케보내는거지"이다.

것이고, 둘째 줄 이하 자소들이 윗 자소들로부터 생성된 자소 후보들이다. 각 자소노드는 자소변환확률값의 로그값인 $\log P(j_{ik} \mid j'_{ik})$[4]를 가지고 있어 $P(C \mid T)$ 계산에 이용된다. 인접한 자소 2개가 후보 생성이 가능할 때에는 그 후보 또한 생성되고[5], 음절 이상 단위의 변환 또한 어절변환패턴 사전으로부터 후보가 생성된다[6]. 그림 1에서 가장 아래의 사각형이 음절 이상 단위의 변환 후보를 나타낸다.

$P(C)$를 계산하기 위해서는 음절단위의 노드가 필요하므로 이 자소후보로부터 다시 음절단위의 후보노드들로의 변환이 이루어진다. 공백문자의 삽입이 이때 이루어져 이 단계에서 후보생성이 완성된다. 생성된 노드로부터 최적 경로를 찾기 위해 Viterbi 방식의 탐색을 맨 왼쪽 노드에서부터 진행하게 된다. 각 음절후보를 선택할 때마다 그 동안 가설마다 누적된 로그 값을 스코어로 생각하여 점수가 낮은 후보들은 beam search를 통해 beam value에서 벗어나는 후보들을 잘라낸다. 그림 2는 최종적으로 생성된 후보의 노드 형태를 나타낸다. 실제로는 후보가 매우 많기 때문에 전체적인 형태를 간단히 나타내었다.

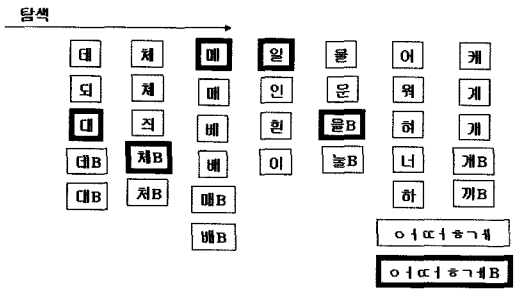

그림 2 최종 후보 생성의 예. 그림 1에서 생성되었던 자소 후보로부터 다음과 같은 음절 후보들이 생성된다. 자소들의 모든 조합이 음절 후보로서 생성될 수 있고, 각 음절에 대해 공백문자 'B'를 붙인 후보들을 다시 생성한다. 어절후보에 대해서도 마찬가지로 공백문자가 붙은 후보를 생성한다. 이제 이 음절 후보들로부터 음절 n-gram을 이용하여 최적경로를 찾아내면 그 경로가 곧 교정 문장이 된다. 탐색은 viterbi 방식으로 이루어지며 각 음절선택시마다 가지치기를 통해 탐색 공간을 줄이게 된다.

4) 실제 구현에서는 파라미터 조정을 통하여 상수 a, b를 결정, "a $\log P(j_{ik} \mid j'_{ik})$ + b"의 값을 이용하였다. (a = 1.0, b = 2.0)

5) Xㅋ → ㅎㅋ, ㅎㅋ, ㄱㅎ, ㅋㅎ, ㄱㅇ

6) ㅇㅓXㅋㅔX → ㅇㅓXㄸㅓㅎㄱㅔX

각각의 음절조합에 대해 후보가 생성되고 그 후보 끝부분에 공백문자 '*B*'가 삽입된 음절들 또한 후보로 들어가게 된다. 여기에서 경로를 찾아나가면 띄어쓰기와 철자 교정이 동시에 이루어질 수 있게 된다. 굵은 선으로 둘러쌓인 사각형이 최적경로로 채택되는 후보 노드들을 나타낸다.

4. 실험 결과 및 분석

4.1 실험 데이타

말뭉치는 gnome irc에서 약 36일 동안 기록된 대화내용이 사용되었다. 여기에서 학습용 말뭉치 78213문장, 실험용 말뭉치 5240문장을 무작위로 선정하였다. 말뭉치들은 모두 수작업으로 띄어쓰기나 철자에서 오류가 발생한 경우를 찾아내어 교정하였고, 따라서 오류가 포함된 말뭉치와 오류가 교정된 말뭉치가 쌍으로 준비되었다. 학습용 말뭉치로부터 얻어진 자소 변환 패턴의 종류는 총 279개, 자소 단위 이외에 음절 단위의 변환 패턴의 종류는 총 1083개였다. 음절 단위의 n-gram(n=1, 2, 3) 또한 학습용 말뭉치로부터 추출되었다. 말뭉치의 자세한 정보는 표 1을 보면 알 수 있다.

표 1 말뭉치 세부 정보. 각각의 항목은 말뭉치에 대해 총 문장 수, 총 어절 수, 총 문장 중 오류 어절이 포함된 문장의 수와 비율, 총 어절 중 오류 어절의 수와 비율을 나타내고 있다.

학습용 말뭉치	문장 수	78213
	어절 수	351815
	오류 문장 수(비율)	21062 (26.93%)
	오류 어절 수(비율)	36643 (10.42%)
실험용 말뭉치	문장 수	5240
	어절 수	21067
	오류 문장 수(비율)	1365 (26.05%)
	오류 어절 수(비율)	1789 (8.49%)

실험은 실험용 말뭉치를 대상으로 하여 실험용 말뭉치에서 철자가 이미 교정된 결과에 띄어쓰기를 단독으로 적용한 결과와 교정이 전혀 되어 있지 않은 실험용 말뭉치에 띄어쓰기와 철자 교정을 병행한 실험결과를 모두 기재하였다. 띄어쓰기 없이 철자 교정만을 하는 실험은 알고리즘 성격상 결과가 잘 나올 수가 없고 실험 자체가 의미가 없을 뿐 아니라 띄어쓰기 교정없이 철자만 교정된 정답 말뭉치가 없기 때문에 실험 결과 평가가 불가능하므로 실행하지 않았다. 또한 각 실험은 기존

의 띄어쓰기 정보를 모두 버리고 음절을 모두 붙인 상태로 실험하였으며, 각각의 실험에 대해서 동일한 조건에서 띄어쓰기 정보를 이용하여 이미 띄어진 공백문자는 무조건 유지하는 형태로 실험을 반복하였다. 이는 텍스트 상에서 띄어쓰기 오류의 대부분은 띄어써야 할 부분을 붙여쓰는 형태이고 붙여써야 할 부분을 띄어쓰는 경우는 거의 없기 때문이다.

4.2 실험 결과 및 분석

먼저 이미 철자가 교정된 말뭉치를 대상으로 띄어쓰기만을 적용한 실험을 하였다. 이는 앞에서 설명한 알고리즘에서 후보 생성 부분을 생략하면 띄어쓰기만을 실행하는 형태가 된다. 실험 결과는 표 2와 같다.

표 2 띄어쓰기 성능 측정

	입력 띄어쓰기 무시	입력 띄어쓰기 유지
정확도(%)	87.88	91.36

띄어쓰기 성능을 어절 단위로 측정한 것이다. 통신어체, 구어체가 중심이 된 본 실험 말뭉치와 다른 논문의 실험에서 쓰였던 말뭉치 자체가 상당히 다르기 때문에 성능의 직접 비교는 힘들지만, 기존에 이루어졌던 한글 문어체를 대상으로 음절 n-gram을 이용하여 띄어쓰기를 실시했던 결과가 어절단위로 91.5%였으며[1], 일본어 OCR인식결과 문장을 대상으로 이루어졌던 실험 결과가 어절단위로 약 92.2%였음을 볼 때[7], 상대적으로 기존의 알고리즘보다 높게 나오지는 않았다. 우리가 제안한 알고리즘은 자소, 어절 단위의 후보 생성 부분을 제외한다면 실질적으로 음절 n-gram을 이용한 [1]과 거의 동일한 방식인데 이 정도의 성능 차이가 나는 것은 실험용 말뭉치와 학습용 말뭉치의 질적, 양적인 차이로 인해 음절 n-gram 언어 모델이 완벽하게 구축이 되지 않았다고 해석할 수 있다. 즉 음절 n-gram의 차이로 인한 성능 저하라는 것인데 이는 나중에 다시 언급할 것이다.

실험용 말뭉치에 대해 제안한 알고리즘을 적용하여 띄어쓰기와 철자 교정을 동시에 수행한 실험 결과는 표 3과 같다. 역시 입력 문장의 띄어쓰기를 유지한 것과 무시한 것 두 가지 상황에 대해 실험하였다.

성능 측정은 전처리기에 의해 교정된 말뭉치와 정답 말뭉치를 어절 단위로 비교, 띄어쓰기와 철자가 전부 일치하였을 경우에 올바로 교정되었다고 판단하여 그 비율을 측정하였다. 일단 가장 눈에 띄는 결과는 교정이 이루어지기 전의 정확도보다 전처리기를 통해 교정된 후의 정확도가 더 떨어진다는 것이다. 이는 다시 말하면 올바른 어절을 교정기가 틀리게 바꾼 경우가 많았다는 것이다. 우리는 이 실험 결과 중 입력 띄어쓰기를 무시

표 3 실험용 말뭉치 전체에 대한 띄어쓰기와 철자 오류 교정 수행 결과. 오류가 교정되기 전의 원본 어절을 raw, 오류가 완전히 교정된 어절을 cor, raw에 교정 시스템을 적용하여 고쳐진 어절을 hyp라고 하였을 때, baseline는 "(raw와 cor가 일치하는 어절 수) / (cor의 총 어절 수)", 실험 결과 정확도는 "(hyp와 cor가 일치하는 어절 수) / (cor의 총 어절 수)"로 정의하였다. 여기에서 어절이 일치한다는 것은 띄어쓰기와 철자가 모두 동일한 경우를 의미한다.

실험 조건	정확도(%)
Baseline(교정 전)	91.51
입력 띄어쓰기 무시	84.57
입력 띄어쓰기 유지	87.09

표 4 실험용 말뭉치 전체에 대한 띄어쓰기와 철자 오류 교정 수행 결과 분석. 오류가 교정되기 전의 원본 어절을 raw, 오류가 완전히 교정된 어절을 cor, raw에 교정 시스템을 적용하여 고쳐진 어절을 hyp라고 하였을 때, "교정없이 일치한 경우"는 raw = hyp = cor, "틀린 어절을 옳게 고친 경우"는 raw ≠ hyp = cor, "틀린 어절을 고치지 못한 경우"는 raw = hyp ≠ cor, "옳은 어절을 틀리게 고친 경우"는 raw = cor ≠ hyp, "틀린 어절을 고쳤지만 또다시 틀린 경우"는 raw ≠ hyp ≠ cor인 경우를 뜻한다.

항목	개수
총 어절 수	23169
교정없이 일치한 경우	16850
틀린 어절을 옳게 고친 경우	1738
틀린 어절을 고치지 못한 경우	347
옳은 어절을 틀리게 고친 경우	3720
틀린 어절을 고쳤지만 또다시 틀린 경우	514

한 결과의 각 어절에 대해 오류 교정이 어떻게 이루어졌는지 분석하였는데 그 결과 실제로 올바른 어절을 틀리게 바꾼 경우가 상당히 많다는 것을 알 수 있었다. 그 분석 결과는 표 4에 나타나 있다.

표 4에서 알 수 있듯이 오류가 포함된 어절을 올바르게 고친 경우보다 옳은 어절을 틀리게 고친 경우가 더 많아서 결과적으로는 정확도가 떨어지고 말았다. 그렇다면 실험용 말뭉치에 포함된 오류 어절의 비율이 증가하였을 때에는 결과는 어떨지 알아보기 위하여 이번에는 앞서의 실험용 말뭉치에서 오류 어절이 포함된 문장 중 2270문장을 추려내어 이를 대상으로 앞서와 같은 과정의 실험을 반복하였다. 그 실험 결과는 표 5에 나타나 있다.

표 5 오류 문장만으로 이루어진 실험용 말뭉치에 대한 띄어쓰기와 철자 오류 교정 수행 결과

실험 조건	정확도(%)
Baseline(교정 전)	73.12
입력 띄어쓰기 무시	81.21
입력 띄어쓰기 유지	83.96

표 5에서 알 수 있듯이 오류 문장만을 대상으로 실험을 하였을 때에는 2가지 경우 모두 baseline을 넘어가는 성능을 보여주고 있다. 역시 이 실험 결과 중 입력 띄어쓰기를 무시한 결과에 대해 오류 분석을 하여 이를 표 6에 나타내었다.

표 6 오류 문장만을 대상으로 한 띄어쓰기와 철자 오류 교정 수행 결과 분석

항목	개수
총 어절 수	13319
교정없이 일치한 경우	8123
틀린 어절을 옳게 고친 경우	2266
틀린 어절을 고치지 못한 경우	501
옳은 어절을 틀리게 고친 경우	1730
틀린 어절을 고쳤지만 또다시 틀린 경우	699

이 경우에는 오류가 포함된 어절을 올바르게 고친 경우가 옳은 어절을 틀리게 고친 경우보다 더 많기 때문에 정확도에서 향상이 이루어질 수 있었다.

결국 앞서의 실험 결과와 이 실험 결과로부터 알 수 있는 사실은 제안된 알고리즘은 정확도(precision)에 취약하다는 것이다. 다시 말하면 false alarm이 매우 많아 잘못된 교정이 이루어졌다는 것이다. 현재 후보 생성 방식으로 거의 모든 오류 어절에 대해 정답이 포함되도록 후보를 생성하는 것이 가능하지만, 후보에서 실제로 최적 경로를 찾는 과정에 있어서 잘못된 경로를 찾는 경우가 많고 이런 현상은 오류가 적게 포함된 문장에 알고리즘을 적용했을 때에 특히 두드러졌다. 이는 후보 생성 과정에서의 적용 범위를 넓히는 것에 성공한 대신 정확도에서 손실이 있었다고 볼 수 있다.

그리고 띄어쓰기 성능이 전체 결과에 상당한 영향을 주는 것을 볼 수 있다. 입력 띄어쓰기를 무시했을 때와 유지했을 때의 성능 차이가 약 3~4% 나고 있는데 입력 띄어쓰기가 잘못된 경우가 있다고 하더라도 그 정보를 이용하는 것이 성능상의 이점이 있다고 볼 수 있다. 이는 앞서 띄어쓰기만의 성능이 매우 높지는 못하다는 점에서 그만큼 기존 띄어쓰기 정보의 이용이 유용했다고도 볼 수 있다. 또한 띄어쓰기 정보를 이용한다는 것은 결국 각각의 어절에 대해 교정과정을 적용한다는 의미이므로 음절 n-gram에 의한 점수 매김보다 자소변환 확률값에 의한 점수매김의 비중이 더욱 컸을 것이다. 이와 같은 조건에서 띄어쓰기 정보를 무시했을 때에 비하여 성능이 높게 나왔다는 것은 음절 n-gram의 신뢰도가 떨어진다는 것을 간접적으로 보여준다고 할 수 있다.

우리는 그 외에도 성능이 낮게 나온 이유를 다음과 같이 예상하고 있다. 일단 말뭉치의 부족을 들 수 있다. 자소나 음절 단위의 변환 확률값은 어느 정도의 말뭉치만으로도 신뢰할 만한 수치를 얻을 수 있고, 실험 결과에서 보듯이 그 확률값을 적당한 상수로 취해도 수치가 많이 떨어지지는 않는다. 하지만 $P(C)$를 계산하기 위해 필요한 음절 단위의 n-gram(n=1, 2, 3) 데이타는 충분한 양의 말뭉치가 있어야 신뢰성 있는 데이타를 얻을 수 있다. 실험에서 발생한 오류들을 분석해보면, 실제로 생성된 후보들 중에 정답이 있었지만 음절 n-gram 스코어에 따라서 그 후보가 잘려나가는 경우가 대부분이었다. 이는 음절 n-gram이 학습용 말뭉치에 매우 의존적이며, 학습용 말뭉치에 포함되어 있지 않았던 어휘들에 대해서는 그 후보를 채택할 확률이 거의 없기 때문이다.

그리고 말뭉치의 오류 태깅이 일관성있게 되어 있지 못하여 성능 측정치가 신뢰성이 높다고 하기 어렵다. 오류 태깅이 모두 2명의 학생에 의해 수작업으로 이루어졌는데, 단순히 철자가 틀린 글자들에 대해서는 태깅이 잘 되어 있지만 통신어나 축약어, 그리고 일반 사람들이 잘 쓰지 않는 말에 대해 표준 태깅 기준이 없기 때문에 오류 태깅을 빠뜨린 경우가 많았다. 태깅을 했더라도 같은 단어에 대해 여러 가지로 교정이 되어 있어서 교정이 올바르게 되었더라도 정답 말뭉치에 나와 있는 그 단어대로 고쳐지지 않았다면 모두 오답처리가 되어 버렸다.

마지막으로 gnome irc 말뭉치는 대화자가 대부분 프로그래머, 개발자라는 특성상 일반 대화체와는 사용된 어휘가 다르며 전문 용어가 많다. 이는 음절 n-gram데이타의 질에도 영향을 준다. 이런 어휘들에 대해 상당히 성능이 낮게 나왔다. 이는 현재 모델이 미등록어나 학습용 말뭉치에서 잘 나타나지 않았던 단어에 대해서 틀린 후보들을 생성하고 그것을 채택할 수밖에 없는 한계점을 보여주기도 한다.

이번 실험은 ME(Maximum Entropy)기법 등을 이용한 정교한 파라미터 최적화 과정 없이 가장 기본이 되는 모델 그대로를 구현하여 실험이 이루어졌기 때문에 개선의 여지가 상당히 크다고 볼 수 있다. 또한 특정 어휘에 편중된 말뭉치와 수작업으로 이루어진 오류 교정 결과의 질 등이 전체 성능을 떨어뜨리고 있다고 생각된다. 이후 좀 더 일반적인 대화체의 말뭉치를 다수 확보하고, 정확히 교정된 말뭉치가 같이 제공된다면 더욱 정

확한 실험 결과를 얻을 수 있을 것이다.

5. 결 론

우리는 텍스트전처리에서 특히 띄어쓰기 오류와 철자 오류를 동시에 교정할 수 있는 새로운 방법을 제시하였다. 자연어 처리 부분에서 많이 쓰이는 noisy-channel model을 기본으로 하여, 공백 문자 생성과 자소 변환을 동시에 할 수 있는 모델로 확장하였고 그 외의 오류들은 어절변환패턴 사전을 이용해 알맞은 후보들을 생성하였다. 통계적 자료인 자소 변환 확률을 이용함으로써 최대한 사전의 크기와 이용률을 줄이면서도 자소 조합으로 인해 올바른 후보가 빠지지 않도록 할 수 있었다. 실험 결과는 이와 같은 방법이 현재로서는 그 성능이 만족스럽지는 않지만 가능성은 충분히 있음을 말해 주고 있다.

앞으로의 연구 방향 중 하나는 모델의 개선에 있을 것이다. 아직 성능이 만족스럽지 못한데 가장 큰 이유는 앞서 표 4에서 볼 수 있듯이 자소 후보의 과생성으로 인하여 교정할 필요가 없는 어절까지 틀리게 고쳐지는 경우(false alarm)가 많기 때문이다. 이를 방지하기 위해서는 자소 모델 생성시에 일정 확률 이상을 가지는 자소만을 후보로 생성하여 과생성을 막는 방법이 있고, 또는 자소 후보 생성 이전 단계에서 오류 감지부를 두어서, 오류가 없다고 판단되는 음절에 대해서는 애초에 자소 후보를 생성하지 않도록 하는 방법이 있을 것이다. 후보 과생성을 막는 구체적인 방법론이 세워져 현재의 모델을 개선할 수 있다면 전체적인 성능의 향상을 기대할 수 있을 것이다. 또 다른 성능 향상의 가능성으로는 단순한 자소 변환 외에 개선된 후보 생성을 통해 이루어질 수 있다. 현재는 단순히 자소의 변환만으로 오류 생성을 다루고 있는데, 자소의 변환시 앞뒤 문맥의 자소 정보를 이용하여 좀 더 구체적인 데이타를 얻을 수 있을 것이다. 또는 최근에 중국어에서 채팅시 이용되는 문장 오류의 대부분을 발음에 따른 변화로 설명하면서 교정하는 모델이 제안되었는데[9] 한국어에도 이와 같은 발음에 따른 후보 생성이 적용 가능하다면 후보 과생성을 줄이면서 동시에 특히 Precision 성능도 개선시킬 수 있다. 또한 학습용 말뭉치에서 나타나지 않았던 미등록어나 고유명사 등을 잘못 교정하는 경우에 대해 어떻게 성능을 향상시킬 수 있을지도 후추 연구에 있어서 중요한 과제가 될 것이다.

참 고 문 헌

[1] 권오욱, "마코프 체인 및 음절 N-그램을 이용한 한국어 띄어쓰기 및 복합명사 분리", 한국음향학회지, 2002, pp. 274-283.

[2] Jianfeng Gao, Mu Li and Chang-Ning Huang, "Improved Source-Channel Models for Chinese Word Segmentation," Proceeding of the 41st Annual Meeting of the ACL, 2003, pp. 272-279.

[3] Christopher C. Yang and K. W. Li, "A Heuristic Method Based on a Statistical Approach for Chinese Text Segmentation," *Journal of the* American Society for Information Science and Technology, 2005, pp. 1438-1447.

[4] Eric Mays, Fred J. Damerau and Robert L. Mercer, "Context Based Spelling Correction," IP&M, 1991, pp. 517-522.

[5] R. L Kashyap, B. J. Oommen, "Spelling Correction Using Probabilistic Methods," Pattern Recognition Letters, 1984, pp. 147-154.

[6] Mu Li, Muhua Zhu, Yang Zhang and Ming Zhou, "Exploring Distributional Similarity Based Models for Query Spelling Correction," Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, 2006, pp. 1025-1032.

[7] Masaaki Nagata, "Context-Based Spelling Correction for Japanese OCR," Proceedings of the 16th conference on Computational linguistics, 1996, pp. 806-811.

[8] Seung-Shik Kang and Chong-Woo Woo, "Automatic Segmentation of Words using Syllable Bigram Statistics," Proceedings of 6th Natural Language Processing Pacific Rim Symposium, 2001, pp. 729-732.

[9] Yunqing Xia, Kam-Fei Wong and Wenjie Li, "A Phonetic-Based Approach to Chinese Chat Text Normalization," Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, 2006, pp. 993-1000.

노 형 종
2001년~POSTECH(학사-CSE). 2006년~POSTECH(석박사통합, 재학중-CSE)

차 정 원
2002년 포항공대 박사. 2002년~2003년 USC/ISI 박사후과정. 2003년~2004년 이화여대 전임강사. 2004년~현재 창원대학교 컴퓨터공학과 조교수. 연구분야는 자연어처리, 정보검색, 기계학습

이 근 배

1984년 서울대학교 컴퓨터공학과 학사 1986년 서울대학교 컴퓨터공학과 석사 1991년 UCLA 컴퓨터학과 박사. 1991년 3월~1991년 9월 UCLA 연구원. 1991년~1996년 포항공과대학교 조교수. 1997년~2003년 포항공과대학교 부교수. 2000년~2001년 미국 Stanford CSLI 연구원. 2004년~현재 포항공과대학교 정교수. 관심분야는 자연언어 처리, 음성인식, 정보검색, 바이오 인포메틱스