음소의 분류 체계를 이용한 한글 편집 거리 알고리즘

(Edit Distance Problem for the Korean Alphabet with Phoneme Classification System)

노 강 호 † 박 근 수 †† 조 환 규 ††† 장 소 원 ††††
(Kangho Roh) (Kunsoo Park) (Hwan-Gue Cho) (Sowon Chang)

요 약 문자열에 대한 편집 거리 문제는 하나의 문자열을 다른 문자열로 변환할 때 필요한 최소한의 연산의 개수를 구하는 문제이다. 영어와 같은 1차원 문자열에 대한 최적해에 대해서는 오랫동안 연구가 진행되어 왔으나, 한글과 같이 좀 더 복잡한 언어에 대한 편집 거리에 대해서는 많은 연구가 진행되지 못했다. 본 논문에서는 음소와 음절을 구분하여 편집거리를 구하는 기존 연구를 확장하여, 음소간의 유사도를 정의하고 이를 이용하여 유사한 단어를 더 정확하게 구분해 내는 알고리즘을 제안한다.

키워드 : 음소 분류 체계, 편집 거리, 한글, 알고리즘

Abstract The edit distance problem is finding the minimum number of edit operations to transform a string into another one. It is one of the important problems in algorithm research and there are some algorithms that compute an optimal edit distance for the one-dimensional languages such as the English alphabet. However, there are a few researches to find the edit distance for the more complicated language such as the Korean or Chinese alphabet. In this paper, we define the measure of the edit distance for the Korean alphabet with the phoneme classification system to improve the previous edit distance algorithm and present an algorithm for the edit distance problem for the Korean alphabet

Key words : edit distance, the korean alphabet, algorithm

1. 서 론

문자열 A와 B에 대한 편집거리는 A를 B로 바꾸기 위하여 필요한 최소한의 연산(삽입, 삭제, 치환)의 개수를 나타낸다. 편집 거리 문제에 대해서는 동적 프로그래밍을 이용하여 $O(|A||B|)$ 시간 안에 최적해를 구하는 여러 방법들이 알려져 있다[1-3].

한글에 대한 편집거리에 대해서는, 음절을 음소 단위로 구분하여 편집거리를 정의하고, 그것을 구하는 알고리즘에 대한 연구 결과가 있다[4]. 논문 [4]에서는 음소를 비교할 때, 서로 다른 두 음소는 모두 같은 거리를 갖는다고 가정하고 있다. 그러나 한글 음소 분류체계를 이용하면, 음소들을 발음할 때의 혀의 높이나 입의 모양 등에 따라 특성이 비슷한 음소들로 이루어진 여러 그룹으로 구분할 수 있으며, 이를 이용하면 한글의 특징을 더 정확히 반영한 편집 거리를 정의할 수 있다. 또한 삽입, 삭제, 치환만을 고려한 편집 거리 알고리즘을 확장하여 음소와 음절이 일치하는 경우까지 고려하면 더 좋은 결과를 얻을 수 있다. 본 논문에서는 한글 음소 분류 체계를 이용한 유사도 점수를 정의하고 그것을 구하는 알고리즘을 제시하였다.

· This work was supported by Korea Research Council of Fundamental Science and Technology.

† 비 회 원 : 서울대학교 전기컴퓨터공학부
kRoh@holybelief.com
†† 종신회원 : 서울대학교 전기컴퓨터공학부 교수
kpark@theory.snu.ac.kr
††† 정 회 원 : 부산대학교 정보컴퓨터공학부 교수
hgcho@pusan.ac.kr
†††† 비 회 원 : 서울대학교 국어국문학과 교수
sowon@snu.ac.kr

논문접수 : 2010년 5월 6일
심사완료 : 2010년 9월 14일

정보과학회논문지: 시스템 및 이론 제37권 제6호(2010.12)

2. 한글 편집 거리에 대한 기존 연구

영어가 알파벳의 1차원 배열 형태를 갖는 것과 달리 한글은 하나 이상의 음소로 이루어진 음절의 배열로 이루어지며, 음소에는 모음(vowel, V)과 자음(consonant, C)의 2가지 종류의 음소가 존재한다. 음절은 자음과 모음의 조합에 따라 여러 형태가 존재할 수 있으나, 본 논문에서는 C, V, CV, CVC의 4가지 형태를 고려하였다.

논문 [4]에서는 3가지의 한글 편집거리 알고리즘을 설명하고 있다. 제일 간단한 알고리즘으로는 음절 단위로 삽입, 삭제, 치환의 3가지 편집 연산을 적용하는 방법을 생각할 수 있는데, 이를 *SylED* (음절 기반의 편집 거리) 알고리즘이라고 칭한다. 다음으로는 음절을 음소 단위로 구분한 후, 대응되는 음소들 간에 존재하는 삽입, 삭제, 치환의 횟수를 계산하는 방법을 생각할 수 있는데, 이를 *PhoED* (음소 기반의 편집 거리) 알고리즘이라고 정의하였다. 다음으로는 이들 알고리즘을 개선하여, 한글의 음절과 음소의 특징을 이용한 편집거리 알고리즘인 *KorED*를 제안하고 있다.

*KorED*는 음소 단위 연산(phoneme edit operation, *PEO*)와 음절 단위 연산(syllable edit operation, *SEO*)을 각각 3가지씩 정의하였다. 먼저 *PEO*는 음소-삽입, 음소-삭제, 음소-치환의 3가지로 정의되며, 각각의 *PEO*에 대한 비용은 모두 α로 정의한다. 음절단위 연산인 *SEO*에 대해서도 음절-삽입, 음절-삭제, 음절-치환의 3가지 종류를 정의하며, 각각의 *SEO*에 대한 비용은 다음과 같이 정의한다. 먼저 음절-삽입과 음절-삭제의 비용은 β로 정의하는데, 이 때 $\beta \geq 3\alpha$가 성립한다. 음절-치환의 경우에는 두 음절을 구성하는 음소들에 따라 2가지 방법으로 정의된다. 먼저 대응되는 모든 음소들이 서로 다른 경우에는 음절-삽입, 음절-삭제와 같은 β를 부여하고, 하나 이상의 음소가 같은 경우에는 $\alpha \times$(서로 다른 음소의 개수)로 정의한다. 이를 정리하면 다음과 같다. 두 음절 A, B간의 거리를 $\delta_S(A,B)$라고 표기하기로 한다.

- Case 1: 음절-삽입: $d_S(\Lambda, B) = \beta$
- Case 2: 음절-삭제: $d_S(A, \Lambda) = \beta$
- Case 3: 음절-치환: $d_S(A,B) = \beta$, A와 B의 대응되는 모든 음소들에 대하여 음소-삽입, 음소-삭제, 음소-치환이 발생한 경우
- Case 4: 음절-치환: $d_S(A,B) = N \times \alpha$, A와 B의 대응되는 초성, 중성, 종성 중에서 최소 하나 이상의 위치의 값이 같은 경우, N은 서로 다른 음소의 개수를 나타냄

본 논문에서는 한국어에서의 음소들을 음소의 특성에 따라 몇 개의 그룹으로 구분한 후, 같은 그룹에 속하는 음소와 그렇지 않은 음소에 대하여 서로 다른 점수를 부여하는 방법을 사용한다. 영어에 대해서도 알파벳이 가지는 특징을 고려하여 알파벳을 여러 가지 그룹으로 구분한 후, 서로 다른 거리를 부여하는 방법이 알려져 있다[5]. 논문 [5]에서는 먼저 알파벳이 가질 수 있는 속성을 자음/모음, 유성음/무성음, 경음/연음과 같은 방법으로 9가지로 구분한 후, 각각의 알파벳이 가지는 속성을 정의하였다. 아래의 표는 알파벳 's'와 'z'가 9가지 속성에 대하여 어느 쪽에 속하는지를 보여주고 있으며, 9가지 속성 중에서 2가지에서만 서로 다른 값을 가짐을 보여주고 있다. 위의 방법을 이용하여 알파벳 간의 유사한 정도를 정의한 후, 유사한 알파벳에는 더 작은 거리를 부여하는 편집 거리 알고리즘을 정의하였다[6].

음소	속성
s	Consonant, Obstruent, Fricative, Continuant, Anterior, Strident, Coronal, Unvoiced, Fortis
z	Consonant, Obstruent, Fricative, Continuant, Anterior, Strident, Coronal, Voiced, Lenis

3. 음소의 특징을 반영한 편집 거리

3.1 한글의 음소 분류 체계

*KorED*에서는 서로 다른 두 음소간의 거리를 모두 동일하게 생각하였다. 예를 들어 'ㄱ'과 'ㄲ'의 거리와 'ㄱ'과 'ㅃ'의 거리를 모두 α로 정의하고 있다. 그러나 한국어의 음운론적 특징을 고려하면, 'ㄱ'은 'ㅃ'보다는 'ㄲ'과 더 밀접한 관계를 갖는다. 한국어 음운론 분야에서 이루어진 음소 분류 체계는 이러한 음소들 간에 존재하는 관계를 정의한 것이다.

먼저 자음의 분류 체계를 살펴보자. 한국어의 자음 분류는 크게 조음 방식과 조음 위치라는 두 가지 기준에 의해서 행해진다. 아래의 표는 조음 위치에 따라 양순음, 치조음, 경구개음, 연구개음, 성문음의 5가지로 구분하는 방법을 보여준다. 조음 방법을 따라 파열음, 마찰음, 유음, 비음 등으로 구분하는 방법이 있으나, 본 논문에서는 첫 번째 구분 방법을 기준으로 편집 거리를 정의하였다.

구 분	초 성	종 성
양순음	ㅁ, ㅂ, ㅃ, ㅍ	ㅁ, ㅂ, ㅄ, ㅍ
치조음	ㄴ, ㄷ, ㄸ, ㅌ, ㄹ, ㅅ, ㅆ	ㄴ, ㄵ, ㄶ, ㄷ, ㄹ, ㄼ, ㄽ, ㄾ, ㅀ, ㅅ, ㅆ, ㅌ
경구개음	ㅈ, ㅉ, ㅊ	ㅈ, ㅊ
연구개음	ㄱ, ㄲ, ㅋ, (ㅇ)	ㄱ, ㄲ, ㄳ, ㅋ, ㅇ
성문음	ㅎ	ㅎ
공백	초성은 공백이 없음	λ

다음에는 모음의 분류체계를 살펴보도록 하겠다. 모음은 혀의 높낮이에 따라서는 고모음, 중모음, 저모음으로 나뉘고, 혀의 전후 위치에 따라서는 전설 모음과 후설 모음으로 나뉘며, 입술의 모양에 따라서는 원순모음과 평순모음으로 나뉜다. 단모음만 제시하면 다음과 같다.

	전설모음		후설모음	
	평순모음	원순모음	평순모음	원순모음
고모음	ㅣ	ㅟ	ㅡ	ㅜ
중모음	ㅔ	ㅚ	ㅓ	ㅗ
저모음	ㅐ		ㅏ	

앞으로는 편의상 위의 분류체계에서 서로 같은 형태로 분류된 것을 그룹이라고 부르도록 하겠다. 서로 같은 그룹에 속한 음소들은 다른 그룹에 속한 음소들보다 더 가까운 관계에 있다고 말할 수 있다. 본 논문에서는 이와 같은 특징을 이용하여 좀 더 한글에 적합한 편집거리를 정의하였다.

3.2 수정된 한글 음소 분류 체계

새로운 편집거리를 정의하고, 실험을 진행하기 위하여 기존 분류체계의 일부를 수정하였다. 먼저 자음 중에서 성문음 'ㅎ'을 연구개음과 같은 그룹으로 편성하였다. 성문음은 'ㅎ' 하나로만 이루어져 있기 때문에, 'ㅎ'과 모든 다른 자음들 간의 거리는 모두 동일한 값을 갖게 된다. 그러나 실생활에서는 '열심히'-> '열심이', '고요히'->'고요이'와 같이 'ㅇ'과 'ㅎ'을 혼동하여 사용하는 사례들을 많이 관찰할 수 있다[7]. 따라서 'ㅎ'을 'ㅇ'과 같은 그룹에 편성하고, 그 이외의 분류 체계는 모두 초성 분류 체계를 그대로 사용하였다.

구분	초 성	종 성
그룹1	ㅁ, ㅂ, ㅃ, ㅍ	ㅁ, ㅂ, ㅄ, ㅍ
그룹2	ㄴ, ㄷ, ㄸ, ㅌ, ㄹ, ㅅ, ㅆ	ㄴ, ㄵ, ㄶ, ㄷ, ㄹ, ㄺ, ㄻ, ㄼ, ㄽ, ㄾ, ㅀ, ㅅ, ㅆ, ㅌ
그룹3	ㅈ, ㅉ, ㅊ	ㅈ, ㅊ
그룹4	ㄱ, ㄲ, ㅋ, ㅇ, ㅎ	ㄱ, ㄲ, ㄳ, ㅋ, ㅇ, ㅎ
그룹5	-	ㅅ

모음은 자음에 비하여 좀 더 많은 수정이 필요하였다. 우선 한국어 음운론에서 시행하는 모음 분류표는 단모음만을 대상으로 하므로 유사계열의 이중모음을 분류대상에 포함시켜서 음소 목록의 수가 증가하였다. 그리고 모음의 그룹화 작업을 위해 몇 가지 기준을 추가로 도입하였다. 우선 동일 그룹에 속한 음소간의 거리값을 최소화하려는 목표 하에 혼동 가능성이 크다고 간주되는 모음들끼리 동일 그룹으로 묶는 작업을 시행하였다. 여기에서 '혼동 가능성이 크다'는 표현은 음운론적 특성이 유사하면 할수록 서로 중화되거나 혼동되어 섞여 사용되는 경향이 크다는 연구 결과에 기반한 것으로, 한국어 음운론 분야에서 이루어진 모음동화 현상에 대한 연구 결과를 고려하였다. 즉, 현대 한국어의 화자들 간에는 특정지역을 제외하고는 'ㅐ'와 'ㅔ'의 음가가 중화되었으므로 표기상으로도 혼동되는 경우가 많다. 다음으로 현대국어에서 발음에서의 경제성을 이유로 일어나고 있는 음운변화 현상을 고려하였다. 예를 들어 'ㅔ', 'ㅚ'의 'ㅣ'로의 단모음화 현상이 일반적으로 이루어지고 있는데, '베다~비다', '희망~히망'의 발음이 구별되지 못하는 것이 그러한 예들이다[8-11].

이와 같은 음운변화를 근거로 그룹1은 'ㅣ, ㅔ, ㅐ, ㅖ, ㅒ, ㅢ'로 구성하였다. 그룹2는 원래 단모음인 'ㅚ', 'ㅟ'가 현대로 오면서 점차 이중모음화하여 기존의 이중모음 'ㅖ', 'ㅙ'와 발음상 구별되지 않는다는 지적을 근거로 한 것이다. 'ㅟ'는 'ㅣ'나 'ㅢ'와의 혼동 가능성을 보아 그룹1에 묶을 수도 있겠지만, 모음 자체가 지닌 원순성을 고려하여 그룹2에 분류하였다. 그룹3에 'ㅕ, ㅑ, ㅖ, ㅒ, ㅝ, ㅘ'를 편성한 이유는 "구멍을 막아."가 "구멍을 막어."로 발음나는 예에서 보듯이 현대국어로 오면서 어말 위치에서 'ㅏ'가 'ㅓ'와 혼동되어 발음되는 경우가 많기 때문이며, 이 두 단모음에 이들과 관련되는 이중모음을 같이 포함시킨 것이다. 그룹4는 'ㅜ'와 'ㅗ'를 묶은 것인데 '삼촌~삼춘', '사돈~사둔'에서처럼 이들 간의 혼동이 많이 일어나기 때문이며 'ㅡ'는 '비늘~비눌'에서처럼 평순모음('ㅡ')이 원순모음화('ㅜ')되는 경향이 강한 음운론적 특성에 근거한 것이다.

구 분	모 음 종 류
그룹1	ㅣ, ㅔ, ㅐ, ㅖ, ㅒ, ㅢ
그룹2	ㅟ, ㅚ, ㅞ, ㅙ,
그룹3	ㅕ, ㅑ, ㅖ, ㅒ, ㅝ, ㅘ
그룹4	ㅡ, ㅜ, ㅗ, ㅠ, ㅛ

3.3 유사도 점수 정의

앞에서 정의한 음소 분류체계를 사용하면, 음소들 간의 거리를 같은 그룹에 속하는 것과 다른 그룹에 속하는 것으로 나누어 2가지로 정의할 수 있다. 기존 알고리즘이 음소와 음절이 서로 다른 경우를 모두 동일하게 처리하였던 것에 비하여, 본 논문에서 제안하는 알고리즘에서는 서로 같은 그룹에 속한 음소와 다른 그룹에 속한 음소에 서로 다른 점수를 부여하게 된다.

먼저 음소에 대한 유사도 점수를 정의하도록 하겠다. 음소 a가 속한 그룹을 $g(a)$라고 표기하도록 한다. 두 음소 a, b가 서로 같으면 이를 음소-match라고 부르며, 유사도 점수는 양수인 M_p를 갖는다. a와 b가 서로 다

른 경우는 음소-mismatch라고 부르며, a와 b가 속한 그룹에 따라 2가지로 정의된다. a와 b가 서로 다른 그룹에 속하는 경우, 즉 $g(a)\neq g(b)$인 경우에는 $-N1_P$를 부여하고, 서로 같은 그룹에 속하면 $-N2_P$를 부여한다. 서로 같은 그룹에 속하는 경우에 더 높은 점수를 주어야 하므로, $-N1_P \leqq -N2_P$의 관계가 성립한다. 음소가 삽입되거나 삭제된 경우는 음소-indel이라고 부르며, 음수 $-I_P$를 할당하였다. 이상의 내용을 정리하면 아래와 같다. 두 음소 a와 b에 대한 유사도 점수는 $\delta_P(a,b)$로 표기하고, 음절 A와 B의 유사도 점수는 $\delta_S(A,B)$로 나타내도록 한다.

- 음소-match : $\delta_P(a,a)=M_P$
- 음소-mismatch : $\delta_P(a,b)=-N1_P,\ g(a)\neq g(b)$
- 음소-mismatch : $\delta_P(a,b)=-N2_P,\ g(a)=g(b)$
- 음소-indel : $\delta_P(a,\lambda)=\delta_P(\lambda,a)=-I_P$

다음에는 음소간의 유사도 점수를 이용하여 음절의 유사도 점수를 정의하도록 하자. 음절도 음소의 경우처럼 match, mismatch, indel의 3가지 경우로 나누어 생각하였다. 음절-match는 음절을 이루는 음소가 모두 같은 경우로 양수인 M_S를 부여하였다. 한글의 특징을 고려할 때, 음절-match인 경우에는 음절을 이루는 음소의 개수에 따라 서로 다른 점수를 부여하는 것보다는 음소의 개수에 상관없이 동일한 점수를 부여하는 것이 좋다고 판단하였다. 즉 δ_S('가','가')와 δ_S('강','강')는 모두 M_S의 값을 갖게 된다. 또한 M_S는 음절을 이루는 각각의 음소의 점수의 합보다는 큰 것이 옳다고 판단하여, 음소-match 점수인 M_P의 3배보다 크거나 같도록 정의하였다.

음절-mismatch는 음절을 이루는 음소의 종류에 따라 2가지 방법으로 정의된다. 먼저 대응되는 음소들이 모두 음소-match가 아닌 경우, 즉 '강'과 '억'과 같은 경우에는 음소 $-N_S$를 부여하였다. $-N_S$는 음절-match와 같은 이유로 음절을 이루는 음소의 개수 관계없이 $-Ns$의 값을 가지며, $-N_S \leqq -3N1_P$의 관계가 성립한다. 대응되는 음소들 중에서 하나 이상이 음소-match인 경우에는, 음절의 유사도 점수를 대응되는 음소의 유사도 점수의 합으로 정의하였다. 예를 들어 '가'와 '남'은 모음이 서로 같으므로, 유사도 점수 δ_S('가','남')은 δ_P('ㄱ','ㄴ')+δ_P('ㅏ','ㅏ')+$\delta_P(\lambda$,'ㅁ')$=-N1_P+M_P-I_P$가 된다.

마지막으로 음절-indel은 음수인 $-I_S$를 부여하였으며, 음절-mismatch와 같은 이유로 $-3I_P$보다는 작도록 정의하였다. 이상의 내용을 정리하면 음절 A와 B에 대한 점수는 아래와 같다.

- 음절-match : $\delta_S(A,A)=M_p \geqq 3M_p$
- 음절-mismatch : $\delta_S(A,B)=-N_p \leqq -3N1_p$, 대응되는 모든 음소들이 음소-match가 아닌 경우
- 음절-mismatch : $\delta_S(A,B)=\sum_{k=1}^{3}\left(\delta_P(A^k,B^k)\right)$, 대응되는 모든 음소들 중 하나 이상에서 음소-match가 발생한 경우. A^1,A^2,A^3은 각각 음절 A의 초성, 중성, 종성을 나타냄
- 음절-indel : $\delta_S(A,\Lambda)=\delta_S(\Lambda,A)=-I_S \leqq -3I_P$

두 문자열 A와 B에 대한 배치 점수(alignment score)는 대응되는 모든 음절의 유사도 점수의 합으로 정의된다. 두 문자열 A와 B간에는 여러 종류의 배치가 존재할 수 있는데, 그 중에서 가장 점수가 큰 배치의 점수를 A와 B의 유사도로 정의하며, 이를 $SIM(A,B)$이라고 표기하기로 한다.

4. 알고리즘

다음에는 두 문자열의 유사도 점수를 구하는 점화식(recurrence relation)을 정의하였다. 두 문자열간의 유사도는 동적 프로그래밍 기법을 사용하여 구할 수 있으며, 그 점화식은 아래와 같이 정의된다. 아래 식에서 $A(i)$는 길이가 i인 문자열 A의 접두사(prefix)를 뜻한다. 이후부터는 이 알고리즘을 $GrpSim$(그룹 기반의 유사도 알고리즘)라고 부르기로 하자[12].

한글 유사도 알고리즘

Base Case

$SIM(\Lambda,B(j))=-j\times I_S$ for $|B|\geqq j\geqq 0$

$SIM(A(i),\Lambda)=-i\times I_S$ for $|A|\geqq i\geqq 0$

General Case

$$SIM(A(i),B(j)) = \max(SIM(A(i),B(j-1))-I_S,\ SIM(A(i-1),B(j))-I_S,\ SIM(A(i-1),B(j-1))+\delta_S(A[i],B[j]))$$

예제 1. '일반통계학'과 '일방통행'간의 유사도 점수를 위의 점화식을 이용하여 구해보았다. 계산을 위하여, $M_S=N_S=I_S=3$, $M_P=N1_P=N2_P=I_P=1$이라고 가정하였다. 아래 표와 같이 점화식을 이용하여 구한 두 문자열의 유사도는 $2M_S+3M_P-2N1_P-N2_P-I_S=3$이 된다.

	ㅅ	일	방	통	행
ㅅ	0	$-I_S$	$-2I_S$	$-3I_S$	$-4I_S$
일	$-I_S$	M_S	M_S-I_S	M_S-2I_S	M_S-3I_S
반	$-2I_S$	M_S-I_S	M_S+2M_P $-N1_P$	M_S+2M_P $-N1_P-I_S$	M_S+2M_P $-N1_P-2I_S$
통	$-3I_S$	M_S-2I_S	M_S+2M_P $-N1_P-I_S$	$2M_S+2M_P$ $-N1_P$	$2M_S+2M_P$ $-N1_P-I_S$
계	$-4I_S$	M_S-3I_S	M_S+2M_P $-N1_P-2I_S$	$2M_S+2M_P$ $-N1_P-I_S$	$2M_S+2M_P$ $-N1_P-N_S$
학	$-5I_S$	M_S-4I_S	M_S+2M_P $-N1_P-3I_S$	$2M_S+2M_P$ $-N1_P-2I_S$	$2M_S+3M_P$ $-2N1_P$ $-N2_P-I_S$

□

5. 실험 결과 및 분석

본 논문에서 제시한 방법이 단어간의 유사성을 적절히 판단함을 알아보기 위하여 두 가지 실험을 진행하였다. 먼저 [4]의 논문에서 사용한 것과 동일한 데이터를 이용하여, 첫 번째 실험을 진행하였다[7]. 이 데이터는 (가까이,가까히), (강낭콩,강남콩), (덥다,더웁다)와 같이 실생활에서 사람들이 헷갈리기 쉬운 689쌍의 단어쌍을 포함하고 있다. 실험을 위해서 이 데이터를 이용하여 다음과 같은 두 가지 종류의 단어쌍의 집합을 구성하였다.

- X: 유사단어의 집합, 위에서 찾은 689개의 단어쌍으로 이루어지며, (가까이,가까히), (강낭콩,강남콩), (덥다,더웁다)등이 X에 포함된다. 이 중에서 (열쇠,쇳대)와 같이 단어의 형태가 완전히 상이한 일부 단어쌍은 제외하였다.
- Y: 비유사단어의 집합, Y는 X에 속한 모든 단어쌍 중에서 첫 번째 단어들만을 추출한 후, 이들 단어들에 대한 모든 조합으로 이루어진 단어쌍의 집합이다. 즉 X에 포함된 단어쌍 중에서 첫 번째 단어인 '가까이', '강낭콩', '덮다'등의 조합인, (가까이,강낭콩), (강낭콩,덮다)와 같은 단어쌍들이 Y에 해당한다.

편집거리 알고리즘을 비교할 때, X에 속한 단어쌍은 유사하다고 판단하고, Y에 속한 단어쌍은 유사하지 않다고 판단하는 비율이 높을수록 좋은 방법이라고 할 수 있다. X에 속한 단어쌍을 유사하다고 판단할 확률을 P_X, Y에 속한 단어쌍을 잘못 판단하여 유사하다고 판단할 확률을 P_Y로 표기하도록 하자.

실험에 사용한 파라미터는 다음과 같다. 먼저 *SylED, PhoED, KorED*에 대해서는 기존 논문에서 제안한, 변수를 그대로 사용하였다. 즉 *SylED*의 경우에는 음절이 다른 경우에 대해서 9를 할당하고, *PhoED*는 음소가 서로 다른 경우에 대해서 3의 값을 부여하였으며, *KorED*에 대해서는 $\alpha=3$, $\beta=9$을 사용하였다.

*GrpSim*에 대해서는 음소와 음절에 대한 match, mismatch, indel에 대한 변수의 값을 정하기 위하여, match, mismatch, indel의 유사도 점수를 나타내는 변수들인 M_S, N_S, I_S, M_P, $N1_P$, $N2_P$, I_P의 값을 각각 0에서 10까지 바꾸어 가면서, 가장 좋은 결과를 보이는 변수값을 찾는 실험을 진행하였다. 평가 기준은 P_X가 96% 이상일 때, 즉 유사 단어를 96% 이상의 확률로 구분해 낼 때의 P_Y가 작을수록 좋은 방법이라고 판단하였다. 다음의 표는 가능한 모든 조합 중에서 좋은 결과를 보였던 5가지 조합을 보여주고 있다. 그 중에서도 M_S, N_S, I_S, M_P, $N1_P$, $N2_P$, I_P이 각각 4, 9, 5, 1, 3, 2, 1일 때에 가장 나은 결과를 보였으며, 유사 단어를 96.03%로 구분해 낼 때의 에러 확률이 0.84%로 매우 정확한 결과를 나타내었다.

실험결과로부터 match, mismatch, indel 중에서 mismatch가 유사도 점수에 가장 큰 영향을 줌을 확인할 수 있었다. 결과가 가장 좋았던 첫 번째 조합을 살펴보면, match 보다는 mismatch에 큰 가중치가 주어짐을 알 수 있다. 음절-match는 4인데 비하여 mismatch는 $-N_S=-9$가 부여되며, 음소는 match는 1인데 비해 mismatch는 두 음소가 같은 그룹이면 -2, 다른 그룹이면 -3이 되어, mismatch에 2배 이상의 가중치가 주어지고 있다. 삽입 또는 삭제인 indel인 경우에는 대략 match와 비슷한 정도의 값을 보여주었다.

음 절			음 소				실험 결과	
M_S	N_S	I_S	M_P	$N1_P$	$N2_P$	I_P	P_X	P_Y
4	9	5	1	3	2	1	96.03%	0.84%
3	9	4	1	3	2	0	96.18%	0.94%
4	9	4	1	3	2	0	96.03%	0.94%
6	9	5	2	3	2	0	96.03%	0.96%
8	9	8	2	3	2	0	96.03%	0.97%

아래의 표는 가장 좋은 결과를 보인 첫 번째 조합에 대한 자세한 실험 결과를 보여주고 있다. *SylED, PhoED, KorED*은 편집거리를 계산하고, *GrpSim*은 유사도를 계산하기 때문에, 각 알고리즘이 나타내는 편집거리와 유사도를 표의 왼쪽과 오른쪽 열에 나타내었다. 즉 제일 왼쪽에 있는 편집 거리는 *SylED, PhoED, KorED*의 결과를 보기 위한 값이며, 제일 오른쪽 열의 유사도는 *GrpSim*에 대한 유사도 점수를 나타낸다. 먼저 *SylED*와 *PhoED*의 실험 결과를 살펴보자. *SylED*에서 편집거리가 18인 경우와 *PhoED*에서 편집거리가 15인 경우는, P_X는 96.2%로 동일하지만, P_Y은 각각 6.9%, 5.2%가 되어, *PhoED*가 *SylED*에 비하여 좋은 결과를 보여줌을 알 수 있다. 비슷한 방법으로 *PhoED*와 *KorED*에서 편집거리가 15인 경우를 보면 P_X는 각

각 96.2%, 95.8%로 거의 동일하지만, P_Y는 각각 5.2%, 3.7%가 되어 *KorED*가 더 나은 결과를 보여줌을 알 수 있다.

다음에는 본 논문에서 제안한 *GrpSim*과 *KorED*의 결과를 비교해보도록 하자. *KorED*의 편집거리가 15인 경우와 *GrpSim*의 유사도가 1인 경우를 보면 P_X는 95.8%와 95.7%로 거의 동일하지만 에러 확률인 P_Y는 *GrpSim*이 0.6%로 *KorED*의 각각 3.7%보다 현저히 좋은 결과를 보여주고 있다. *KorED*의 편집거리가 9인 경우와 *GrpSim*의 유사도가 9인 경우에도 P_X는 89.1%와 89.0%로 거의 동일하지만, P_Y는 각각 0.3%, 0.1%로 *GrpSim*이 더 좋은 결과를 보여주었다.

편집 거리	*SylED*		*PhoED*		*KorED*		*GrpSim*		유사도
	P_X	P_Y	P_X	P_Y	P_X	P_Y	P_X	P_Y	
0	0	0	0	0	0	0	88.1	0.1	10
1	-	-	-	-	-	-	89.0	0.1	9
2	-	-	-	-	-	-	89.0	0.1	8
3	-	-	69.4	0	68.4	0	89.3	0.2	7
4	-	-	-	-	-	-	89.7	0.2	6
5	-	-	-	-	-	-	91.9	0.3	5
6	-	-	83.5	0.1	82.6	0	91.9	0.3	4
7	-	-	-	-	-	-	92.1	0.3	3
8	-	-	-	-	-	-	94.1	0.4	2
9	77.9	0.3	89.6	0.3	89.1	0.3	95.7	0.6	1
10	-	-	-	-	-	-	95.7	0.6	0
11	-	-	-	-	-	-	95.9	0.7	-1
12	-	-	94.2	1.5	93.8	1.1	95.9	0.7	-2
13	-	-	-	-	-	-	96.0	0.8	-3
14	-	-	-	-	-	-	96.2	1.2	-4
15	-	-	96.2	5.2	95.8	3.7	96.3	1.8	-5
16	-	-	-	-	-	-	96.3	2.3	-6
17	-	-	-	-	-	-	96.6	2.9	-7
18	96.2	6.9	98.4	13.0	98.0	10.0	97.5	4.8	-8
19	-	-	-	-	-	-	97.5	5.2	-9

다음에는 인터넷 등에서 볼 수 있는 비속어에 대한 실험을 진행하였다. 온라인에서는 검열에 의하여 많은 비속어가 사용될 수 없기 때문에, 검열을 피할 수 있는 변형된 비속어들이 많이 사용된다. 이러한 단어들은 검열을 피하기 위하여 의도적으로 만든 단어들이기 때문에, 일부 음소의 순서가 바뀌거나, 그와 유사한 다른 음소로 대체되는 경우가 많다. 예를 들어 비속어 '미친새끼'의 유사 단어로는 '미친새퀴', '미친색기'등이 이에 해당될 수 있다.

실험 데이터는 한국콘텐츠진흥원의 비속어 데이터를 이용하였다. 이 데이터는 욕설 단어와 그것의 변종 단어들을 포함하고 있는데, 이를 이용하여 (욕설,변종단어) 형태의 단어쌍으로 이루어진 데이터를 만들었다. 데이터를 만들 때에, 욕설 단어가 숫자나 특수 기호가 포함하고 있거나, '개새ㄲㅣ'와 같이 하나의 음절이 여러 개의 음절로 나뉘어 있는 형태는 포함시키지 않았다[13].

알고리즘에 대한 평가는 첫 번째 실험과 비슷하게 진행하였다. 먼저 욕설 데이터의 단어쌍들을 특정 비율로 유사하다고 판별하는 편집거리 또는 유사도 점수를 구한 후, 그에 따른 실험 데이터 1에서의 P_Y값을 살펴보았다.

아래의 표는 욕설 데이터를 약 92%와 96%로 구분해 낼 때의 P_Y값을 보여주고 있다. 먼저 욕설 데이터를 90%로 구분해 낼 때의 결과를 보면, 욕설 데이터를 구분해내는 값은 모두 92% 정도로 비슷하였으나, P_Y는 *GrpSim*이 2.5%로 *PhoED*나 *KorED*의 6.2%, 4.1%에 비하여 현저히 좋은 결과를 보여주었다. 욕설 데이터를 96%로 구분해내는 경우에도 *GrpSim*의 P_Y가 6.3%로 다른 알고리즘의 47.0%, 14.3%, 12.1%에 비하여 매우 좋은 결과를 보여주었다.

구 분		*SylED*	*PhoED*	*KorED*	*GrpSim*
92%정도로 구분할 때	구분 확률	-	92.9	92.0	92.4
	P_Y	-	6.2	4.1	2.5
96%정도로 구분할 때	구분 확률	96.8	97.5	96.6	96.8
	P_Y	47.0	14.3	12.1	6.3

6. 결 론

본 논문에서는 기존 논문에서 제안한 한글 편집거리를 개선한 새로운 편집거리를 제안하였다. 이를 위하여 먼저 한글 음소 분류체계를 이용하여 음소 간에 존재하는 유사성을 적용하였다. 그리고 기존 논문이 음소, 음절의 삽입, 삭제, 치환만을 고려한 것에 비하여, match인 경우까지를 고려한 유사도 점수를 정의하였다.

새로운 알고리즘은 기존의 알고리즘보다 더 좋은 결과를 보여주었다. 유사 단어에 대한 실험에서는 유사단어가 아닌 단어를 유사단어로 잘못 판단할 확률이 1/3이하로 줄어들었으며, 비속어 데이터에 대한 실험에서도 에러 확률이 절반 가까이 감소하였음을 확인할 수 있었다.

참 고 문 헌

[1] Gusfield, D.: Algorithms on strings, trees, and sequences : computer science and computational biology, Cambridge Univ. Press, January 2007.

[2] Wagner, R. A., Fischer, M. J.: The String-to-String Correction Problem, *J. ACM*, vol.21, no.1, pp.168-173, 1974.

[3] Navarro, G.: A guided tour to approximate string

matching, *ACM Computing Surveys*, vol.33, no.1, pp.31-88, 2001.
[4] Kangho Roh, Jin Wook Kim, Eunsang Kim, Kunsoo Park, Hwan-Gue Cho, Edit Distance Problem for the Korean Alphabet, *Journal of KIISE*, 2010 (in korean).
[5] M. Pucher, A. Turk, J. Ajmera, N. Fecher, Phonetic Distance Measures for Speech Recognition Vocabulary and Grammar Optimization, 3rd congress of the Alps Adria Acoustics Association, Graz, Austria, September 2007.
[6] Gong, R., Chan, T. K.: Syllable Alignment: A Novel Model for Phonetic String Search, IEICE - Trans. Inf. Syst., E89-D(1), 2006, 332-339.
[7] Sowon Chang, Seong-kyu Kim, Seung-chul Jung, This slip of the tongue that slip of the pen: official documents, Ministry of Culture and Tourism, 2000. (in korean).
[8] Young-Seon Kim, Research on Assimilation and Syllabification of Korean, 1999, Sŏul-si : Kukhak Charyowŏn. (in korean).
[9] Ki-Moon Lee, Korean Phonology, Hakyensa, 2006. (in korean)
[10] Jin-Ho Lee, Lectures on Korea phonology, Samkyungmunhwasa, 2005. (in korean)
[11] Sung Mun Cho, Principles and Restrictions for the Phoneme Phenomenon of Korean Consonant, Hankookmunhwasa, 2000. (of korean)
[12] Hirschberg, D. S.: A linear space algorithm for computing maximal common subsequences, Commun. ACM, vol.18, no.6, pp.341-343, 1975.
[13] http:/www.kocca.kr

노 강 호
정보과학회논문지 : 시스템 및 이론
제 37 권 제 2 호 참조

박 근 수
정보과학회논문지 : 시스템 및 이론
제 37 권 제 1 호 참조

조 환 규
정보과학회논문지 : 시스템 및 이론
제 37 권 제 2 호 참조

장 소 원
1984년 서울대학교 인문대학 국어국문학과 학사. 1986년 서울대학교 대학원 국어국문학과 석사. 1991년 프랑스 파리 제5대학교 언어학박사. 현재 서울대학교 인문대학 국어국문학과 교수. 관심분야는 텍스트언어학, 한국어문법론