■ 2010년도 학생논문 경진대회 수상작

제한된 한글 입력환경을 위한 음소기반 근사 문자열 검색 시스템

(A Phoneme-based Approximate String Searching System for Restricted Korean Character Input Environments)

윤 태 진 † 조 환 규 †† 정 우 근 †
(Taijin Yoon) (Hwan-Gue Cho) (WooKeun Chung)

요 약 모바일 기기가 발전함에 따라 입력 수단에 대한 연구는 중요한 이슈이다. 키패드, 쿼티키패드, 터치, 음성인식 등 다양한 입력장치가 사용되고 있으나 아직 데스크톱 입력장치에 비해 편의성이 떨어져서 입력 시의 오타나 탈자 등의 오류가 포함되는 경우가 많다. 이러한 입력 오류는 문자 메시지 등 사람과의 의사소통에는 문제를 일으키지 않으나 사전, 주소록 등의 데이터베이스 검색에는 치명적인 오류로서 원하는 검색 결과를 얻지 못하게 된다. 특히 한글의 경우 자음과 모음의 조합을 통해 글자를 생성하는 특성상 1만자가 넘는 글자의 조합이 가능하여 영문에 비하여 오류의 빈도가 높다. 기존의 검색 시스템은 Suffix Tree등을 이용하여 입력 오류를 처리하지만 다양한 오류에 대응하기에는 한계가 있다. 본 논문에서는 오자, 탈자 등의 입력 오류를 허용하면서 빠른 검색이 가능한 근사 한글 단어 검색 시스템을 제안하고자 한다. 이 시스템은 기존의 알파벳에 적용된 근사 문자열 검색(Approximate String Searching)을 한글에 효과적으로 적용할 수 있는 여러 가지 알고리즘과 기법이 포함되어 있다. 그리고 제안된 시스템을 이용한 변형 욕설 필터링 시스템의 개발에 대해 이야기하고자 한다. 이 시스템은 유저의 각종 변형 욕설 입력에 대해 90% 이상의 필터링 성능을 보였다.

키워드 : 한글 문자열, 근사 문자열 검색, 전역 정렬

Abstract Advancing of mobile device is remarkable, so the research on mobile input device is getting more important issue. There are lots of input devices such as keypad, QWERTY keypad, touch and speech recognizer, but they are not as convenient as typical keyboard-based desktop input devices so input strings usually contain many typing errors. These input errors are not trouble with communication among person, but it has very critical problem with searching in database, such as dictionary and address book, we can not obtain correct results. Especially, Hangeul has more than 10,000 different characters because one Hangeul character is made by combination of consonants and vowels, frequency of error is higher than English. Generally, suffix tree is the most widely used data structure to deal with errors of query, but it is not enough for variety errors. In this paper, we propose fast approximate Korean word searching system, which allows variety typing errors. This system includes several algorithms for applying general approximate string searching to Hangeul. And we present profanity filters by using proposed system. This system filters over than 90% of coined profanities.

Key words : Hangeul string, approximate string matching, global alignment

· 본 연구는 2010년도 연구재단 일반연구자지원사업(2010-0015665)의 연구비 지원으로 수행하였습니다.

† 학생회원 : 부산대학교 정보컴퓨터공학부
ytj@pusan.ac.kr
wkchung@pusan.ac.kr
†† 정 회 원 : 부산대학교 정보컴퓨터공학부 교수
hgcho@pusan.ac.kr

논문접수 : 2010년 6월 1일
심사완료 : 2010년 8월 4일

Copyright©2010 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.
정보과학회논문지: 소프트웨어 및 응용 제37권 제10호(2010.10)

1. 서 론

누구나 한 두 글자의 오타로 인해 원하는 검색결과를 얻지 못하는 경험을 가지고 있을 것이다. 특히 모바일 기기의 경우 키패드나 터치스크린과 같은 제한되고 불편한 입력 수단을 사용하게 되므로 데스크톱의 경우보다 오타를 입력하게 될 확률이 높다. 불편한 입력 장치 때문에 어렵게 입력한 검색어가 오타로 인해 다시 입력해야 할 경우의 스트레스는 이루 말할 수 없을 정도이다. 운전 중에 급하게 내비게이션을 조작하다가 오타로 인해 잘못된 결과가 출력 되었을 때의 스트레스는 모두 공감할 것이다. 오타가 단순한 조작 미스로 인한 것이 아니라 잘못된 기억으로 인해 발생했을 경우 문제는 더욱 심각해진다. 사회가 복잡해짐에 따라 한 개인이 관리해야 하는 데이터량은 점점 증가하고 있다. 정렬과 검색 시스템의 도움으로 효율적인 데이터의 관리가 이루어지고 있으나 검색에 필요한 키워드(Keyword)는 어디까지나 사람의 기억에 의존할 수밖에 없다. 그러나 사람의 기억은 불완전하여 때때로 검색에 필요한 정확한 키워드를 도출해내지 못하는 경우가 발생한다. 특히 한글의 경우 이 문제에 더욱 취약하다. 한글은 자음과 모음의 조합으로 1만개가 넘는 글자가 존재하고 그 중에는 유사한 발음의 단어가 여럿 존재한다. 또한 적혀 있는 글자와 발음의 차이가 나는 경우가 많아 검색에 필요한 정확한 키워드를 입력하는데 어려움이 있다. **"바람"**과 **"바램"**, **"찌개"**와 **"찌게"** 등은 사람들이 쉽게 착각하는 대표적인 예이다. 표 1은 틀리기 쉬운 프로펠러의 오기 사례이다.

표 2와 같이 상호명은 틀리기 쉬운 단어의 대표적인 예이다. 필자의 경험을 예로 들자면 친구의 추천을 받아 "석대추어탕"이라는 맛집을 찾기 위하여 내비게이션의 상호 검색 기능을 사용하여 검색하였다. 그러나 검색결과는 0이었다. 그래서 결국 친구에게 들은 위치를 바탕으로 한참을 헤매서 찾아갔는데 도착해서 간판을 보니 간판 위에 "원조"라는 글자가 붙어있었고 내비게이션에도 "원조석대추어탕"이라고 등록되어 있었다. 내비게이션 시스템에서는 첫 글자 부터 완전히 일치하지 않으면 제대로 된 검색결과를 보이지 않았던 것이다. 실제로 사람들이 상호 등을 기억할 때 핵심되는 부분, 즉 "석대"라는 부분을 기억하지 "원조"와 같은 수식 단어까지 기억하는 경우는 드물다. 요즘 휴대폰의 필수 기능이 된 초성 검색의 경우 이런 문제를 어느 정도 방지할 수 있으나 초성 역시 정확하게 입력되지 않으면 정확한 검색결과를 기대하기 어렵다. "신쭈꾸미"라는 음식점을 검색하고자 하면 "ㅅㅉㄲㅁ"라고 정확하게 입력해야 하지 "ㅉㄲㅁ"나 "ㅅㅈㄱㅁ" 등 정확하지 못한 검색에 대해서는 올바른 결과를 보이지 않는다.

표 1 '프로펠러'에 대해 자주 볼 수 있는 오기 사례 - 모바일 입력 장치에서 유사한 발음의 글자는 이웃해 있거나 같은 버튼을 공유하는 경우가 많아 오타가 나올 확률이 높다.

정상단어	오기 사례
프로펠러	프로팰러 프로페라 프로펠라 프로페러 프로팰라 프로패라 프로패러

표 2 원래 단어와 사람에게 기억되는 단어의 차이 - 사람은 단어를 기억할 때 핵심적인 부분만 요약해서 기억하는 습관이 있다. 단어의 앞이나 뒤에 추가적으로 붙는 수식어의 경우 기억에 남아있지 않을 가능성이 높다.

등록된 상호명	일반적으로 기억되는 상호명
원조석대추어탕	석대추어탕
신쭈꾸미	쭈꾸미
부산중앙저축은행	중앙저축은행
늘 편한 내과	편한 내과
베럴 디자인 쇼파	배럴 쇼파

이것은 근사 문자열 검색 문제이며 중요한 Open Problem의 하나로 예전부터 많은 연구가 이루어져 오던 분야이다. 영어권 국가에서는 이미 많은 연구가 이루어져 Alphabet을 위한 근사 단어 색인(Approximate String Indexing)은 Suffix Tree[1], 거리공간(Metric Space)와 다차원 자료구조[2] 등을 이용하여 빠른 검색이 가능하게 하였다. 그러나 한글은 알파벳과는 표기 방식에 큰 차이가 있어 알파벳에 적용된 방법을 그대로 사용할 경우 원하는 결과를 얻기가 어렵다. 알파벳 1글자와 한글 1글자는 포함하고 있는 정보의 양과 비중에서 많은 차이를 보이기 때문이다. 단순한 예를 들자면 "감"이라는 한글을 알파벳으로 표기하기 위해서는 "gam"이라는 3글자가 필요하다. 그리고 알파벳 26자에 대해서 한글은 초성, 중성, 종성의 조합을 통해 1만자가 넘는 글자의 표현이 가능하다.

본 논문에서는 기존의 근사단어 검색 시스템을 효율적으로 한글 단어에 적용시키기 위한 여러 방법이 적용된 최적화된 한글 근사 단어 검색 시스템에 대해서 설명할 것이다. 우리는 이미 생물정보학적인 분석기법을 이용하여 한글 문서간의 유사도를 측정하는 DeVAC (Document eVolution Analysis Center)연구를 해왔다[3-5]. 이 시스템은 한글로 쓰인 두 문서간의 유사도를 BLAST(Basic Local Alignment Search Tool)[6]를 사용하여 효율적인 분석이 가능한 시스템이며 어느 문서가 어느 문서를 표절하였는지 표절의 방향에 대한 정보 역시 제공해준다. 비록 DeVAC은 두 문자열 간의 유사도를 측정하는데 매우 강력하고 빠른 시스템이나

단어와 단어 혹은 단어와 데이터베이스간의 유사도를 분석하기에는 어렵다. 우리는 이 DeVAC을 개발해왔던 노하우를 바탕으로 한글의 입력 시에 발생할 수 있는 다양한 오류를 효과적으로 대처하여 사용자가 원하는 검색 결과를 보이는 시스템을 개발하였다. 편집 거리가 거리공간을 만족하는 성질을 이용하여 다차원 자료구조를 통한 고속 근사 문자열 검색 시스템으로 보다 사람의 인지형태에 가까운 단어의 유사성 판정이 가능하도록 하였다. 이를 보다 효과적으로 만들기 위해 실험을 통해 시스템의 최적화를 이루어 내었다. 더불어 이 근사 한글 단어 검색 시스템은 인터넷의 발달로 날로 문제가 심각해지고 있는 언어 폭력문제를 해결할 수 있는 비속어 필터링 시스템에 적용될 수도 있다. 우리는 근사 한글 단어 검색 시스템과 유전자 정보를 분석할 때 주로 사용되는 서열 정렬 알고리즘[7]을 이용하여 기존의 고정적인 금칙어 필터링 시스템으로는 걸러낼 수 없는 변형 비속어 필터링 시스템을 개발하였다. 이 시스템은 기존의 시스템이 거의 걸러내지 못하는 변형 욕설을 90% 이상 걸러낼 수 있다. 예를 들어 게임 언어 건전화 지침서에 실린 금칙어 리스트로는 "썩을넘"의 변형인 "써ㄱ을넘"을 필터링 할 수 없지만 본 시스템에서는 무리 없이 필터링 할 수 있다.

2. 근사 문자열 검색에 관한 연구들

근사 문자열 매칭문제는 이미 많은 연구가 이루어진 분야로 다양한 알고리즘이 연구되었으며 다양한 해결 방법이 검증되어 있다. 가장 일반적인 방법으로는 편집거리인 Hamming Distance를 이용하여 기존의 1차원적인 정렬을 벗어나서 실제 인간의 단어 인식 방법에 가까운 형태로 tree나 다차원 공간을 이용하는 방법이 주로 사용된다[2,8]. 그 외에도 빠른 속도의 검색이 가능한 Compressed Suffix Array, Invert Index 등을 이용한 근사 문자열 검색 시스템의 연구도 이루어지고 있다[9,10]. 본 시스템에서는 편집거리와 다차원 자료구조인 R*tree를 이용한 고속 근사 문자열 검색 시스템을 사용한다. 본 시스템에 적용되는 3가지 알고리즘 3가지 핵심 알고리즘인 단어와 단어, 단어와 데이터베이스 그리고 그 데이터베이스를 이루는 핵심 자료구조에 대해서 설명하도록 하겠다.

2.1 서열 정렬과 편집 거리

근사 단어 검색을 위해서는 먼저 단어와 단어 간의 1:1관계에서 서로간의 유사도를 평가할 수 있는 방법이 필요하다. 기존의 검색 시스템에 사용되는 방식은 단어와 단어가 완전 일치하거나 첫 글자부터 일정 글자까지 일치하는 부분이 존재하지 않으면 단어 간의 유사도를 알 수 없기 때문이다. 예를 들어 "근사단어검색"이라는

```
Global FTFTALILLAVAV
       F--TAL-LLA-AV

Local  FTFTALILL-AVAV
       --FTAL-LLAAV--
```

그림 1 전역정렬과 지역정렬의 차이 - 서열 정렬 방식은 사용되는 목적에 따라 여러 가지 방법이 존재한다. 전역정렬은 규모가 유사한 문자열 간의 분석에 유용하고 지역 정렬은 한 문자열에 다른 문자열이 속해 있는지 여부를 판별할 때 유용하다.

단어를 검색한다면 완전히 일치하는 단어 혹은 "근사단어매칭"과 같은 단어는 유사도의 평가가 가능하나 "유사단어검색"과 같은 단어는 전혀 검색하지 못하기 때문이다. 문자열간의 유사도를 평가하는 방법은 그림 1과 같이 크게 두 가지 방법으로 서열 정렬 방법과 편집 거리가 주로 사용된다.

서열정렬은 문자서열 간의 유사성을 비교하는데 아주 유용한 방법으로 특히 DNA 서열의 전체 유사성, 국부 유사성을 찾는데 사용되고 있다. 우리는 이 방법을 원용하여 한글단어를 자소단위로 풀어 나열하고 그것의 주어진 단어의 DNA 서열과 같이 간주하여 그 자소단위의 스트링을 정렬(Alignment)하여 그들 간의 상호 유사성을 찾아내고자 한다.

전역 정렬(Global Alignment)은 서열 정렬 방법의 한 가지로 잔여 부분 없이 모든 문자에 대하여 정렬을 수행하는 방법으로 길이가 유사한 문자열간의 비교에 유용하게 사용된다. 지역 정렬(Local Alignment)는 길이 차이가 큰 문자열 간의 정렬을 수행할 때 유용한 방법으로 한쪽의 패턴이 다른 문자열의 안에 포함되어 있을 때 사용하는 방식이다.

편집거리는 한 문자열이 다른 문자열과 같아지기 위해서 필요한 최소한의 수정 횟수를 의미한다. 문자열 간의 유사함을 구체적인 수치로 표현할 수 있어 근사 문자열 매칭(Approximate String Matching)을 위한 거리색인(Metric Indexing)의 좌표를 계산하기 위하여 사용된다.

2.2 근사 문자열 매칭을 위한 거리색인(Metric Index)

실제 검색 시스템에서는 단어와 단어 간의 1:1관계의 유사도 계산만으로는 끝나지 않고 데이터베이스내의 수많은 단어와 입력된 검색어간의 일대다 관계에서의 유사도 검색이 필요하다. 서열 정렬 혹은 편집거리를 입력 단어와 데이터베이스 내의 모든 단어와 계산할 경우 많은 연산량이 필요하여 시스템의 성능이 저하된다. 그러므로 1차적인 분류를 통해서 검사할 대상을 줄여줄 필요가 있다. 우리는 이 문제를 해결하기 위하여 "A metric index for approximate string matching"의 알고리즘을 사용한다[11].

거리공간은 삼각부등식(Triangle Inequality)를 만족하는 Black-box Obeject의 집합과 그들 간의 함수이다. 정규적으로 거리공간은 (X, d)의 쌍으로 X는 Object의 전체 집합이고 d : X × X → R+은 거리 함수로 양수만 반환한다. 이 거리는 Reflexivity(d(x, x) = 0, strict Postiveness(x ≠ y → d(x, y) > 0), Symmetry (d(x, y) = d(y, x)) 그리고 삼각부등식(d(x, y) ≤ d(x, z) + d(z, y))을 만족한다.

X의 부분집합 U는 우리가 찾고자 하는 Object의 집합이다. 거리공간의 많은 질의(Query)중에 우리가 주목하는 것은 범위검색(Range Query): 주어진 Query q ∈ X, 허용 반지름 r, U는 q에서 r거리 안에 있는 원소의 집합이다. 정규적으로 질의의 출력은 (q, r)_d = {u ∈ U, d(q, u) ≤ r}이다.

거리공간을 색인(Index)하는 방법은 크게 2가지로 나뉜다. 첫째 pivot기반의 기술로 하나의 일반적인 아이디어로 구성된다. U에서 k개의 원소를 선택하여 {p_1,..., p_k} 각 원소 u ∈ U를 k차원의 점(d(u, p_1),...,d(u, p_k))으로 인식하는 것이다. 이 정보를 가지고, 삼각부등식을 이용해 어떤 원소 u를 어떤 pivot pi에 대해서 |d (q, pi) - d(u, pi)| > r로 필터링할 수 있다. 그 경우 우리는 d(u, q)를 평가하지 않고도 d(q, u) > r이라는 것을 알 수 있다. 이 규칙을 이용해 필터링하지 못한 원소들은 q와 직접적으로 비교하게 된다. 더 많은 pivot을 사용할수록 더 많은 원소가 제거 되지만 색인에 더 많은 메모리를 필요로 하게 되고 좌표 계산에 더 많은 계산량을 요구하게 된다. 효율적인 범위 검색 방법으로는 kd-tree, R*tree, R-tree등에 포함된 방법이 있다[12,13].

2.3 R*tree를 이용한 범위검색

효과적인 근사 단어 색인을 위해서는 우수한 다차원 자료구조가 필요하다. 우리는 다수의 pivot과 단어와의 편집거리(Edit distance)를 이용한 다차원 거리공간을 저장할 자료 구조로 R*tree를 선택하였다[12].

R*tree는 다차원 데이터 처리에서 가장 널리 사용되는 데이터 구조이다. R*Tree는 R-tree[13]의 발전된 형태로 R-tree는 우리가 흔히 사용하는 B-tree를 다차원 정보 Indexing에 맞게 개량한 것이다. 그림 2와 같이

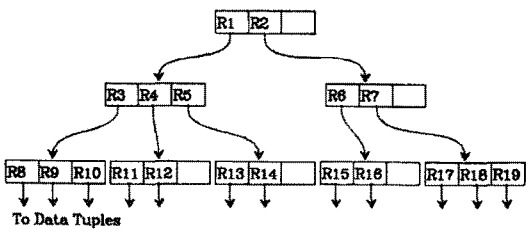

그림 2 R-tree의 구조

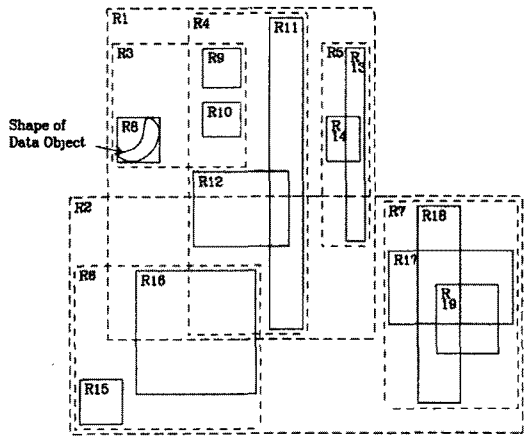

그림 3 R-tree의 MBRs

기본적인 구조는 B-tree와 같이 삽입과 삭제가 일어나더라도 균형을 유지할 수 있는 방법을 취하고 있다.

R-tree는 다차원의 space를 그림 3과 같이 Minimum Bounding Rectangles로 나눈다. 각 노드는 가변적인 수의 원소를 가진다. Leaf노드 외에는 자식노드의 주소와 모든 원소를 포함하는 MBR의 정보를 가지고 있다. 삽입(Insertion)과 삭제(Deletion)은 가장 가까운 MBR을 이용하여 이루어진다.

R*tree는 이 R-tree의 MBR의 빈 공간을 최소화하기 위하여 원소가 삽입될 때 MBR의 원소가 Overflow를 일으킬 경우 MBR의 중심에서 가장 먼 원소를 Reinsertion하여 공간효율을 높이는 방법이다. 공간효율이 높아지면 Overlap이 줄어들어 자료구조의 성능이 높아지게 된다.

3. 대용량 한글 DB를 위한 한글 근사 단어 색인

우리는 서열 정렬 값을 측정하는 방식으로 두 단어 간의 유사도를 측정한다. 그러나 실제 근사 단어 검색을 시스템에 적용할 경우 데이터베이스내의 무수한 단어와 입력단어를 모두 검사하는 것은 지나치게 긴 수행시간을 요구하게 되어 사용이 어렵다. 특히 모바일 환경의 경우 CPU의 연산속도가 제한되므로 데스크톱이나 서버의 경우보다 적은 연산량의 알고리즘이 필요하다. 그러므로 검사를 수행할 단어를 줄여줄 필요가 있다. 검사를 수행할 유사단어 후보군을 추출하기 위하여 앞에서 언급한 근사 문자열 매칭을 위한 거리 색인을 한글에 적합하게 적용하는 방법과 한글에 적합한 서열 정렬 값을 측정하는 방법에 대하여 설명하겠다.

3.1 시스템 개요

본 시스템은 근사 단어 검색 시스템 중에서 데이터베이스의 자료를 사전 처리를 통해서 거리공간에 저장시켜

다차원 자료 구조의 범위 질의(Range Query) 알고리즘을 이용하여 근사 단어를 찾아내는 방식을 취하고 있다. 편집거리를 좌표 값으로 이용하는 이 방식은 편집거리 값이 거리공간을 만족하므로 검색에 대해 올바른 결과 값이 포함되는 것은 보장되어 있으며 이미 다차원 자료 구조는 많은 연구를 거듭하여 R*tree라는 효율적인 자료구조를 사용할 수 있으므로 공간내의 좌표를 결정하는 방식에 따라 시스템의 성능이 결정된다고 할 수 있다.

앞서 설명한 바와 같이 편집 거리는 삼각 부등식을 만족하므로 거리공간의 좌표로 사용하는데 적합한 척도이다. 즉 올바른 질의와 범위가 입력된다면 원하는 단어가 검색되는 것을 보장할 수 있다. 단어를 R*tree에 저장하기에 앞서 이 편집 거리를 측정할 기준이 되는 단어들인 pivot의 선택이 필요하다. 이 pivot의 선택 방법은 검색 시스템의 성능에 큰 영향을 미치므로 다음 장에서 자세하게 다루도록 하겠다. pivot의 선정이 끝나면 데이터베이스의 각 단어를 차례로 입력하면서 pivot 모두와 측정한 편집거리를 다차원 좌표로 사용하여 R*tree에 입력하게 된다.

R*tree에서 저장된 오브젝트인 단어를 검색하기 위해서는 좌표와 검색 반경이 포함된 범위 질의가 필요하다. 좌표의 경우 데이터베이스의 단어를 자료구조에 저장할 때와 마찬가지로 각 pivot과 입력단어 간의 편집거리를 측정하여 검색 목적에 맞는 검색 반경을 입력하여 Hyper Rectangle형태의 검색 질의를 구성한다. R*tree는 해당 Hyper Rectangle에 포함되는 모든 단어를 반환하여 사용자는 입력단어와 유사한 단어를 얻을 수 있게 된다. 사용 목적에 따라 검색 반경을 조절할 수 있으므로 사용자는 적정한 수의 근사 단어를 얻을 수 있으므로 방대한 데이터량에 따른 속도 문제를 해결할 수 있게 된다.

3.2 대표형 변환을 통한 색인 단순화

근사 한글 단어 색인 방법에서 가장 큰 문제는 한글의 글자 수 이다. 자음과 모음을 조합하여 1만개가 넘는 글자를 생성할 수 있는 한글은 다양한 발음의 표현이 가능한 만큼 다양한 입력 오류를 발생시킬 수 있고 색인이 지나치게 세분화 되어서 약간의 오류만으로도 전혀 다른 위치에 색인될 수 있다. 앞서 자음과 모음을 분리시키는 방법으로 상당부분 단순화 시키는데 성공하였으나 한글은 여전히 많은 수의 글자를 가지고 있다.

앞에서 예를 든 "도리"와 "또리", "오리"를 보자. 자소를 분리할 경우 "ㄷㅗㄹㅣ", "ㄸㅗㄹㅣ", "ㅇㅗㄹㅣ" 세 단어 모두 편집거리 1의 거리에 위치하게 된다. 그러나 앞에서 설명하였듯이 "도리"와 "또리"는 더 가까운 위치에 자리하는 것이 합리적인 방법이라고 할 수 있다. 우리는 이 문제를 해결하기 위하여 대표형 변환이라는 방법을 사용하였다.

표 3 효율적인 색인을 위한 대표형 변환 규칙

초 성			
원래 문자	표준화 문자	원래 문자	표준화 문자
ㄱ, ㄲ, ㅋ	ㄱ	ㅅ, ㅆ	ㅅ
ㄷ, ㄸ, ㅌ	ㄷ	ㅈ, ㅉ, ㅊ	ㅈ
ㅂ, ㅃ, ㅍ	ㅂ		
중 성			
원래 문자	표준화 문자	원래 문자	표준화 문자
ㅏ, ㅑ	ㅏ	ㅗ, ㅛ	ㅗ
ㅐ, ㅒ, ㅔ, ㅖ, ㅙ, ㅚ, ㅝ, ㅞ	ㅐ	ㅜ, ㅠ	ㅜ
ㅓ, ㅕ	ㅓ	ㅟ, ㅢ, ㅣ	ㅣ
종 성			
원래 문자	표준화 문자	원래 문자	표준화 문자
ㄱ, ㄲ, ㅋ	ㄱ	ㅂ, ㅍ	ㅂ
ㄷ, ㅌ, ㅅ, ㅈ, ㅉ, ㅊ	ㄷ		

대표형 변환은 유사한 발음과 형태를 가지는 글자를 통합하여 대표 문자로 변환하는 방식을 말한다. 표 3은 본 시스템에서 사용하는 대표형 변환 규칙의 일부이다. 본 표의 규칙에 따라 변환된 문자열을 색인에 활용하게 된다. 'ㄷ'과 'ㄸ'이 같은 'ㄷ'으로 변환되므로 "도리"와 "또리"는 거리공간에서 같은 위치에 존재 하게 되므로 어느 쪽을 근사 문자열 검색하더라도 다른 한쪽을 반드시 포함되게 된다.

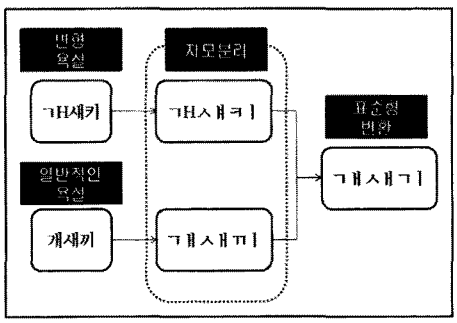

그림 4 대표 형태 변환 과정

그림 4의 대표형 변환은 색인에 사용되는 문자수를 줄여 줌으로써 색인에 사용되는 좌표를 간략화 할 수 있다. 다차원 데이터 구조를 이용하여 근사문자열검색을 할 때 지나치게 듬성듬성한 형태로 Mapping된다면 범위 검색을 하더라도 유사한 단어가 근처에 위치하지 않아 지나치게 넓은 검색 반경을 요구하게 되므로 검색 시간 복잡도가 늘어나게 된다. 더불어 좌표의 차원을 줄일 수 있어 차원의 수가 늘어남에 따라 발생할 수 있는 차원의 저주와 같은 여러 가지 문제에 대해서 방지할 수 있다.

3.3 pivot 단어의 선택 방법

pivot은 근사 문자열 색인에서 성능에 가장 큰 영향을 주는 요소 중 하나이다. 해당 데이터에 맞는 적절한 pivot을 선택해주지 않는다면 올바른 좌표 설정이 이루어지지 않아 범위 검색이 제대로 이루어지지 않게 된다. 예를 들어 너무 적은 수의 pivot을 사용할 경우 데이터베이스의 단어들이 적절하게 분포되지 못하여 검색의 결과로 너무 많은 단어를 반환하여 사용자가 원하는 단어를 찾기가 어려워진다.

검색 결과로 비정상적으로 많은 수의 검색 결과가 반환되는 것을 막기 위해서는 적절한 수의 pivot이 필요하다. 다양한 패턴의 pivot을 좌표로 사용할수록 단어가 고르게 분배되어 적절한 수의 결과를 반환하게 되어 원하는 검색 결과를 얻기가 쉬워진다. 그러나 지나치게 많은 pivot을 사용할 경우 pivot과 입력단어간의 편집 거리를 계산하는 시간이 길어져서 검색 속도 저하를 가져오는 문제점이 있다. 즉 pivot의 수에 따른 검색의 정확도와 속도는 서로 상충관계에 있다고 할 수 있다. 더군다나 pivot의 수가 늘어나면 차원의 수가 같이 늘어나게 되므로 차원의 저주 역시 피할 수가 없게 된다. 효과적인 탐색을 위해서는 데이터베이스의 데이터 량에 따라 적절한 수의 pivot을 결정할 필요가 있다.

적정한 수의 pivot을 선택하기 위해서는 실험을 통한 수치 해석이 필요하다. 데이터베이스에 사용되는 언어, 평균 단어의 길이, 데이터베이스의 목적에 따라 수많은 변인이 존재하기 때문이다. 확실한 것은 pivot의 증가에 따라 단어의 분포가 세분화된다는 것과 반면에 pivot 숫자가 증가함에 pivot숫자에 따른 세분화 효율이 감소하게 된다는 것이다. 본 시스템에서는 후보군 추출 뒤에 후보군과 입력단어와 전역 정렬 값을 측정하기 때문에 pivot의 숫자가 일정 수 이상에 도달할 경우 pivot의 증가가 오히려 검색 효율을 떨어뜨리게 되는 것이다.

pivot의 길이 역시 중요한 요인이 된다. pivot의 길이가 너무 짧을 경우 분포를 세분화하는데 도움이 되지 못하며 너무 길 경우에는 한 pivot이 여러 가지 패턴을 포함하여 오히려 세분화하지 못하는 경우가 생긴다. 예를 들어 "가나다라마바"라는 pivot을 선택할 경우 "가나다"와 "라마바" 두 단어 모두 같은 편집 거리를 가지게 되므로 단어의 분포가 오히려 한곳에 편중되는 현상이 일어나게 된다. pivot의 길이가 길 경우 입력단어의 좌표 측정 시에도 시간이 오래 걸려 검색 효율을 떨어뜨리게 된다. 그리고 패턴과 마찬가지로 pivot의 길이 역시 다양하게 분포되어 있어야 효율이 높다고 할 수 있다. 이것 역시 데이터 분포의 효율성을 높여주기 때문이다.

pivot의 길이 이외에도 pivot의 생성 방법도 상황에 따라 나뉠 수 있다. 가장 보편적인 방법으로는 데이터베이스 내의 단어를 pivot으로 사용하는 것이다. 실제 데이터베이스 내의 단어를 사용함으로써 데이터베이스의 언어와 특성에 맞는 좌표 설정이 가능한 방법으로 가장 효율이 높은 방식이다. 그러나 데이터베이스의 단어가 미리 결정되어 있지 않다면 효과적인 pivot선택이 어렵다는 단점이 있다. 내부의 데이터가 자주 변할 경우 그에 따라 다시 pivot을 선정하여 데이터베이스를 구성한다면 매우 비효율적인 방법이 된다. 이것을 위하여 한글에 사용할 수 있는 범용 pivot을 미리 만들어 둘 수 있다. 이것은 개인이 사용하는 주소록 등의 데이터가 자주 변경되는 시스템에 유용할 것으로 생각된다. pivot을 고정된 몇가지 글자를 선정하여 사용하므로 대부분의 단어에 대해 일정한 성능을 보일 필요가 있다. 그러기 위해서는 가장 보편적인 한글 단어의 집합이라고 할 수 있는 국어사전의 단어를 사용한다. 데이터베이스 내의 단어를 사용하는 것보다 효율이 떨어지기는 하나 최소한의 성능은 보장이 가능하기 때문이다.

구체적인 수치적인 특성은 뒤에 나오는 실험챕터에서 실험의 분석을 통하여 분석하도록 하겠다. 여러 가지 변인 중에서 중요한 파라메터를 제외한 나머지를 통제하여 pivot의 수, 길이, 구성하는 방법에 따른 성능의 변화를 살펴보도록 하겠다.

3.4 한글 음소 기반의 단어 유사도 계산 모델

앞서 설명하였듯이 두 문자열 사이의 1:1 유사도를 판단하기 위해서는 서열 정렬 알고리즘을 사용한다. 그러나 이 서열 정렬은 유전체나 알파벳과 같은 1글자가 1자리를 차지하는 형태의 문자체계에는 유용하나 한글과 같은 초성, 중성, 종성이 합쳐져서 하나의 글자를 만드는 형태의 문자체계에서는 그 적용 방법을 달리할 필요성이 있다. 예를 들어 표 4에서처럼 "간암"과 "가남"은 유사한 발음의 단어이나 반 전역 정렬 값을 측정하였을 때 "가루"가 "간암"보다 "가남"과 유사도가 높게 나온다.

이 문제는 비슷한 발음의 글자사이에 그 발음을 이루는 자소 중 하나라도 차이가 나면 서로 다른 글자가 되기 때문에 일어나는 일이다. 그러므로 이 문제를 해결하기 위해서는 합쳐진 자소를 분리시키는 방법을 사용하

표 4 한글 서열 정렬의 두 가지 측정법

글자 단위		자소단위	
비교 대상	유사도	비교 대상	유사도
가루 간암	-2.0	ㄱㅏㄹㅜ ㄱㅏㄴㅇㅏㅁ	0.0
가남 간암	-2.0	ㄱㅏㄴㅏㅁ ㄱㅏㄴㅇㅏㅁ	4.7
가루 가남	0.0	ㄱㅏㄹㅜ ㄱㅏㄴㅏㅁ	0.0

게 된다. 자소가 분리된 글자 간의 반 전역 정렬을 측정할 경우 "ㄱㅏㄹㅜ-ㄱㅏㄴㅏㅁ"의 경우 0.0이고 "ㄱㅏㄴㅇㅏㅁ-ㄱㅏㄴㅏㅁ"의 경우 4.7이다. 완성된 글자 단위로 반 전역 정렬을 측정하는 경우 보다 발음의 유사성에 가까운 결과를 보이는 것을 알 수 있다.

자소 단위로 글자를 분리하더라도 문제가 모두 해결되는 것은 아니다. 예를 들어 "도리"와 "또리", "오리"를 보자. 자소를 분리할 경우 "ㄷㅗㄹㅣ", "ㄸㅗㄹㅣ", "ㅇㅗㄹㅣ"로 어떤 쌍을 선택하더라도 반 전역 정렬 값은 2.0이다. 그러나 우리는 "도리"와 "또리"의 발음이 "오리"와는 달리 가깝다고 느끼고 실제 오타가 나더라도 전자의 경우가 발생하기 쉽다는 것을 알고 있다. 그러므로 전자를 더 높은 유사도로 측정하는 것이 합리적이라 할 수 있다.

표 5는 본 시스템에서 사용되는 오류 자소에 대한 매칭값(Matching Value)의 일부이다. 이 매칭 값은 인터넷 용어에서 사람들이 주로 사용하는 단어 변형 방법을 조사하여 그 빈도를 기준으로 만들어 졌다. 동일한 자소를 1.0으로 두고 'ㄷ'과 'ㅌ'는 0.8의 매칭 값을 가진다. "만득이"를 자모 분리 시켰을 때 "ㅁㅏㄴㄷㅡㄱㅇㅣ"의 8개의 자소로 최대 8.0의 값을 가지는데 "만특이"를 정렬할 경우 "ㅁㅏㄴㅌㅡㄱㅇㅣ"는 7.8의 매칭값을 가지게 되므로 상대적 유사도는 7.8/8.0 = 97.5%로 계산된다.

표 5 유사 자소의 매칭 값(Matching Value)

기본 자소	매칭자소	점수
ㄱ	ㄲ, ㅋ	0.8
ㄷ	ㄸ, ㅌ	0.8
ㅐ	ㅒ,ㅔ,ㅖ,ㅙ,ㅚ,ㅞ,ㅞ	0.8
ㅣ	ㅟ,ㅢ	0.8

표 6 오류 단어의 반 전역 정렬(Semi-global Alignment) 예

	단어	Alignment 결과
Source	찌개	ㅉ ㅣ ㄱ ㅐ
Targe	지게	ㅈ ㅣ ㄱ ㅔ
Score	3.6	0.8 1.0 1.0 0.8
Source	도리	ㄷ ㅗ ㄹ ㅣ
Targe	또리	ㄸ ㅗ ㄹ ㅣ
Score	3.8	0.8 1.0 1.0 1.0

표 6은 여러 오류 단어들의 반 전역 정렬(Semi-global Alignment) 결과 값을 보여주고 있다. 몇몇 글자에 오류가 있어 다른 단어가 입력되더라도 충분히 높은 유사도가 측정되는 것을 볼 수 있다. 모바일 입력환경에서는 특히 자음을 같은 버튼의 연속 입력을 통해 센소리와 된소리를 처리하는 경우가 많아 본 방식을 이용할 경우 입력오류를 상당히 줄일 수 있다.

4. 응용 : 인터넷 욕설 필터링 시스템

Swear Filter[14]를 기초로 하는 기존의 고정적인 금칙어를 필터링하는 비속어 필터는 수많은 변형 욕설에 대해서 제대로 필터링이 이루어지지 않아 효과가 의문시되었다. 여러 포탈사이트에서는 금칙어 리스트를 확충하는 방식으로 유저의 비속어 사용을 제한하려고 하였으나 결과는 좋지 않았다.

위의 표 7은 유명 게임포탈의 채팅시스템을 이용하여 200개의 변형 비속어에 대한 필터링 테스트 결과이다. 모두 30% 이하의 필터링 성능을 보였는데 사용자의 의도적인 변형 비속어 사용을 거의 막을 수 없다는 것을 알 수 있다. 우리는 이 문제를 해결하기 위하여 앞에서 설명한 근사 한글 단어 검색 시스템과 서열 정렬 알고리즘을 통한 단어 간의 유사도 측정방법을 이용하여 효과적인 변형 욕설 필터링 시스템을 개발하였다.

표 8은 실제 실험에 사용된 단어의 실험 예이다. 내용의 전부가 걸러지지 않고 일부만 걸러지는 경우가 많아 표시된 부분을 통해서 해당 비속어의 추측이 가능하여 필터링의 효과가 그다지 높지 않은 것을 볼 수 있다. 변형 비속어 전체를 필터링하기 위해서는 정확한 근사 한글 단어 검색 시스템이 필요하다.

표 7 한글 변형 비속어 on-line test 결과

	검출된 단어 수	sensitivity
게임포탈(N사)	60	30.0%
게임포탈(H사)	23	11.5%
게임포탈(P사)	20	10.0%

표 8 실제 실험단어의 예. O는 제시된 단어 전체가 걸러진 것을 나타낸다.

입력단어	N사	H사	P사
시 팔년	시 팔년	**년	**년
시발 년	** 년	** 년	** 년
시발새끼	0	0	0
시발새캬	0	**새캬	**새캬
시발쉐리	**쉐리	0	**쉐리
시발	시발	0	시발
시부랄	시부랄	시부랄	시**
시팔년	**년	**년	**년
시팔새끼	0	0	0
시팔새리	**새리	**새리	**새리
십새	0	0	십새
십새끼	0	0	십**
십세끼	십**	0	십세끼

4.1 기존 시스템의 문제점

욕설 필터링 시스템은 이미 여러 분야에서 실제 사용

되고 있다. 그러나 서론에서 언급한 문제를 완전히 해결하지는 못하여서 성능이 높다고는 할 수 없다. 한국 게임산업진흥원에서는 2009년 봄에 "게임언어 건전화 지침서 연구"를 통해서 대표형 2,308항, 총 8,508개의 금칙어 리스트를 작성하여 배포하였으나 위의 두 가지 문제를 이유로 네티즌들의 반응은 부정적이었다[15]. 이 연구의 의미는 실제 사용되는 인터넷 욕설을 총망라했다는 점에서 큰 의미가 있으나 인터넷 욕설의 진화속도를 볼 때 새로운 욕설은 매우 빠르게 생산되고 있어 위와 같은 종합적인 욕설탐색 작업이 그 속도를 따라잡기에는 여러 어려움이 있다.

일반적으로 온라인 게임 채팅 시스템에 사용되는 욕설 필터링 시스템은 Swear Filter[14]가 쓰인다. 입력된 문장을 검색해서 부적합한 단어가 검색되면 입력을 거부하거나 해당 단어를 다른 단어로 변형시켜서 출력하는 방법이다. 간단하고 효율적인 방법이나 변형 욕설을 필터링 하지 못하고 정상단어를 필터링 하기도 하여 실용성은 그렇게 높다고 하기 어렵고 오히려 사용자들의 불만을 사고 있다. 국내 주요 온라인 게임포탈에서도 이 방법을 응용한 자체적인 욕설 필터링 기법을 사용하고 있으나 만족할 만한 성능을 보이고 있지 않다. Shekhar Dhupelia가 언급했듯이 변형 욕설에 대해서는 매우 취약한 방식이기 때문이다[16].

스팸 필터링(SPAM Filtering) 분야에서는 문서에 포함된 단어를 이용한 문서 분류 기법을 사용하여 스팸(Spam)을 걸러낸다. 주로 Naive Bayes, SVM, K-nearest neighbor 등의 기법이 사용되는데 기계 학습(Machine Learning) 방법을 통해서 스팸에 주로 사용되는 단어를 학습시킨 후 그 데이터를 이용하여 스팸을 걸러내는 방식을 사용한다[17]. 한국에서는 "SVM을 이용한 온라인게임 비속어 필터링 시스템"에서 이러한 방식을 사용한 비속어 필터링 시스템을 제안하였다[18]. 그 외에도 e-mail 주소나 서버의 신뢰도를 이용한 스팸 필터 방식도 연구되고 있다[19].

자연어 처리기법을 이용한 비속어 필터링 방식도 생각해 볼만하다. 비유적인 내용을 사용한 문장의 저속한 의미를 형태소 분석 등을 통하여 파악하는데 도움을 줄 수 있다. 그러나 온라인 용어의 특성상 문법과 맞춤법이 지켜지지 않고 특히 욕설을 사용할 때 단어 그 자체를 극단적으로 변형시켜서 사용하는 경우가 많기 때문에 일반적인 자연어 처리 기법으로는 이러한 변형 단어와 문법에 어려움을 겪게 된다. (주)아이모션에서 출원한 특허 "음절결합 정보를 이용한 음란/비속어 차단시스템"에서 자연어 처리기를 기반으로 한 비속어 처리기를 제안하였으나 이 역시 중심음절 그 자체가 변형되어 버리면 올바른 동작을 보장할 수 없게 된다[20].

4.2 한글 근사 단어 검색 시스템을 이용한 후보군 추출

서열 정렬을 이용하여 입력된 단어와 일정 수치 이상의 유사도를 가진 금칙어가 존재하는지 여부를 판정하기 위해서는 먼저 한글 근사 단어 검색 시스템을 이용하여 검사를 수행할 후보군을 추출하여야 할 필요가 있다. 입력단어를 수많은 금칙어와 모두 비교할 경우 필요한 연산량이 너무 많아져 채팅시스템에 지연 현상을 유발하게 되어 쾌적한 커뮤니케이션을 방해할 수 있기 때문이다.

표 9 발음을 이용한 변형 예

기본 단어	변형 단어
개새끼	개새기, 개새뀌, 개색기, 개색휘, 개색히, 개세이, 개쉐리, 개쉐이, 개색
개놈	개늄, 개눔, 개늠, 개넘, 개너므, 게 넘, 게넘, 게놈, 게늄, 게눔, 게늠, 궤놈
병신	뱅쉰, 뱅쉔, 벵신, 벵틴, 병시인, 뷩신, 뷩 쉰, 븅신, 븅쉰, 비융신
씨팔	쉬발, 쉬밸, 쉬뱅, 쉬벌, 쉬벧, 쉬볼, 쉬불, 쉬빌, 쉬파, 쉬팍, 쉬팔, 쉬팕, 쉬팡
불알	붕알, 부랄, 부럴, 브랄, 브라알, 블알, 뿡알, 뿡알

표 9는 발음을 이용한 변형 비속어의 예이다. 앞서 설명한 근사 한글 단어 검색시스템을 사용하면 이러한 발음을 이용한 비속어에 대해서는 대부분 대응이 가능하다. 그러나 일반적으로 오타와 잘못된 기억으로 입력되는 검색어 오류와는 달리 변형비속어는 고의적으로 이루어지는 경우가 많으므로 추가적으로 고려해야할 사항이 존재한다.

변형 비속어 필터링 시스템에서 추가적으로 고려해야할 사항은 일반적인 오류가 아닌 의도적으로 특수문자와 외래어를 사용한 변형이다. 일반적으로 검색 시스템을 이용하면서 외래어나 특수문자가 오타로 들어갈 일은 매우 적으나 비속어의 경우 필터링 시스템을 피하면서 모욕적인 의미를 전달하기 위해 한글 자소와 형태와 발음이 비슷한 알파벳이나 특수문자를 사용하는 경우가

표 10 외래어 및 특수문자를 이용한 변환 규칙

자 음			
원래 문자	표준화 문자	원래 문자	표준화 문자
g, k, c	ㄱ	b, v, p, f	ㅂ
n, L	ㄴ	s, A	ㅅ
d, t, E	ㄷ	o, w	ㅇ
l, r	ㄹ	j, g, z	ㅈ
m, ㅁ	ㅁ	h	ㅎ
모 음			
원래 문자	표준화 문자	원래 문자	표준화 문자
a	ㅏ	u	ㅜ
@, H	ㅐ	I, i, y, !, ㅣ	ㅣ

표 11 변형 욕설의 대표형 변환 예

변형욕설		욕설의 표준형
1	ㄱHㅅHㄲㅣ	ㄱㅐㅅㅐㄱㅣ
	깨쉐끼	ㄱㅐㅅㅐㄱㅣ
2	미친년	ㅁㅣㅈㅣㄴㅕㄴ
	미쳔년	ㅁㅣㅈㅣㄴㅕㄴ
	ㅁI친년	ㅁㅣㅈㅣㄴㅕㄴ
3	씨팔	ㅅㅣㅂㅏㄹ
	쉬팔	ㅅㅣㅂㅏㄹ
	쉬빨	ㅅㅣㅂㅏㄹ
4	쌍년	ㅅㅏㅇㄴㅕㄴ
	쌍ㄴㅕㄴ	ㅅㅏㅇㄴㅕㄴ
5	자지	ㅈㅏㅈㅣ
	자ji	ㅈㅏㅈㅣ

많다. 표 10은 특수문자와 알파벳에 대한 대표형 변환규칙의 일부이다. 만약에 사용자가 "개자식"을 변형하여 입력하여 "ㄱHza식"으로 입력하더라도 위의 변환 규칙을 사용하면 "ㄱㅐㅈㅏㅅㅣㄱ"으로 변환되어 정상적으로 금칙어 리스트를 검색할 수 있게 된다.

4.3 서열 정렬을 이용한 유사도 측정

근사 한글 단어 검색을 이용하여 우리는 입력 단어와 가장 유사한 비속어 집단을 대용량의 데이터베이스에서 추출해내는데 성공하였다. 입력된 변형 욕설이 우리의 데이터베이스에 있는 욕설인지를 판별하기 위해서 반 전역 정렬을 이용한 패턴 매칭 방법을 통해 단어 간의 유사도를 분석하여 일정 수치 이상의 유사도를 보인다면 해당 입력 단어를 비속어로 판정한다. 반 전역 정렬은 전역 정렬과는 달리 입력 단어의 전체와 비교 대상 단어의 일부를 사용해서 정렬을 수행한다. 그러므로 반 전역 정렬은 유사한 욕설을 찾는데 우수한 성능을 보여준다고 할 수 있다. 여기에 특수문자와 알파벳 등을 이용한 변형 비속어에 대응하기 위하여 이를 포함한 Matching Matrix를 추가하여 이들 단어에 대한 매칭 값을 가산할 수 있도록 하였다.

표 12는 본 비속어 필터링 시스템에서 사용되는 비속어 변형에 주로 사용되는 유사 문자 간의 매칭 값의 일부이다. 변형 비속어에 사용되는 변형 형태를 추가

표 12 변형 욕설에 대응하기 위한 매칭 값

기본 자소	매칭 자소	점수
ㄱ	ㄲ, ㅋ	0.8
	g, k, c	0.6
	>	0.4
ㄴ	ㄻ	0.6
	L	0.5
ㄷ	ㄸ, ㅌ	0.8
	d, t	0.6
ㄹ	l, r	0.6

표 13 욕설의 반 전역 정렬 예

	욕설	Alignment 결과
Source	ㅅㅣ입팔	ㅅ ㅣ ㅇ ㅣ ㅂ ㅍ ㅏ ㄹ
Target	씨입팔	ㅆ ㅣ ㅇ ㅣ ㅂ ㅍ ㅏ ㄹ
Score	7.8	0.8 1.0 1.0 1.0 1.0 1.0 1.0 1.0
Source	시이방	ㅅ ㅣ ㅇ ㅣ ㅂ ㅏ ㅇ
Target	쒸이방	ㅆ ㅟ ㅇ ㅣ ㅂ ㅏ ㅇ
Score	6.4	0.8 0.6 1.0 1.0 1.0 1.0 1.0
Source	ㅅㅣ입팔	ㅅ ㅣ ㅇ ㅣ ㅂ ㅍ ㅏ ㄹ
Target	씨이팔	ㅅ ㅣ ㅇ ㅣ _ ㅍ ㅏ ㄹ
Score	6.0	1.0 1.0 1.0 1.0-1.0 1.0 1.0 1.0

로 조사하여 외래어와 유사 형태의 특수문자 등을 추가하여 다양한 형태의 변형 비속어 필터링에 대응할 수 있도록 하였다. 이 Matching Matrix는 추가적인 연구를 통해 통계적으로 많이 사용되는 형태의 변형을 추가하여 더 효율적인 시스템을 구성할 수 있을 것이다.

표 13은 여러 욕설들의 반 전역 정렬 결과값을 보여주고 있다. 몇몇 글자를 변형 시켜서 변형 형태를 만들더라도 충분히 높은 유사도가 측정되는 것을 볼 수 있다. 본 시스템에서는 상대 유사도 75%를 비속어의 기준으로 설정하였는데 세 경우 모두 상대 유사도 75%이상으로 측정되어 비속어로 판정되는 것을 알 수 있다.

4.4 다단계 필터링을 통한 정상 단어 처리

앞서 설명한 Scunthorpe Problem[21]은 욕설 필터링 시스템의 발전을 저해하는 아주 중요한 문제이다. 욕설을 원천 봉쇄하기 위해 강도 높은 필터링을 수행하게 된다면 필터링 되는 정상 단어의 빈도도 높아지기 때문이다.

변형 욕설의 경우 이 문제가 더 민감하게 작용하게 된다. 일반 욕설보다 변형 욕설이 일반 단어와 유사한 형태일 가능성이 높기 때문이다. 특히 반 전역 정렬을 이용하여 유사도 측정을 통해 변형 욕설을 검색하는 본 시스템의 특성상 욕설과 유사한 일반 단어는 높은 측정치를 보여 필터링 될 가능성이 높다. 인터넷 용어는 은어와 신조어가 많이 사용되고 외래어 등의 유입도 빠르기 때문에 이러한 단어들이 필터링 되는 경우 또한 매우 빈번하게 일어날 것이다.

문제의 해결책은 정상 단어를 모아놓은 사전을 이용하여 미리 정상적인 단어를 검증하고 나머지 부분에 대해서만 욕설 필터링을 수행하는 것이다. 그림 5에서는 이러한 문제를 해결하기 위한 순서도를 나타내고 있는데 예를 들어 "시발점"이라는 단어를 입력했을 때 "씨발"이 금칙어 리스트에 포함되어 있다면 "씨발"과 "시발"은 같은 표준형인 "ㅅㅣㅂㅏㄹ"로 변환되고 유사도도 높기 때문에 욕설 필터링에 의해 걸러지게 된다. 그러나 "시발점"이라는 단어를 미리 정상 단어 사전에 추가하고 단어를 미리 검증해 둘 경우 정상적으로 단어를 입력할 수 있다.

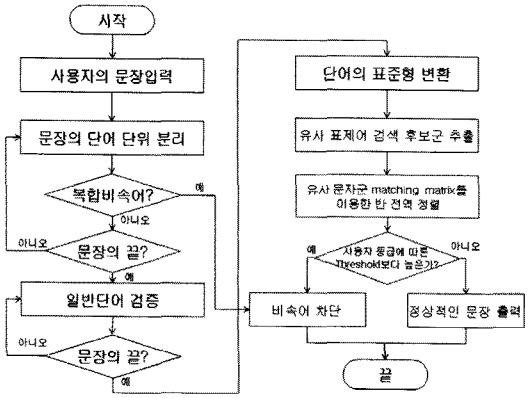

그림 5 다단계 욕설 필터링 시스템

반면에 정상 단어 검증은 욕설 필터링과 같은 부작용을 일으키게 된다. 욕설 필터가 정상 단어를 욕설로 필터링하는 경우처럼 정상 단어 필터가 욕설을 정상 단어로 필터링하는 경우이다. "말뻐다귀"의 경우 "말"라는 단어와 "뻐다귀"라는 단어의 복합어이다. 이 단어를 그냥 일반단어 검증을 하고 욕설 필터링을 한다면 "말뻐다귀"는 정상 단어 두개로 인식되어 욕설 필터링을 빠져나가게 된다.

이러한 부작용을 막기 위한 해결책은 정상 단어 검증 방법과 유사하다. 정상 단어 검증 전에 욕설 필터링을 한 번 더 거치는 것이다. 이 욕설 필터링은 변형 욕설 필터링과는 달리 명백히 욕설이라 할 수 있는 것만 필터링하여 Scunthorpe problem이 일어나지 않도록 하여야 한다.

5. 실 험

이 장에서는 시스템의 성능을 측정하기 위해 크게 2가지 실험을 수행한다. 첫째, pivot의 파라메터와 선택 알고리즘에 따른 검색 효율성실험이다. 근사 검색 시스템에서 가장 성능에 크게 영향을 미치는 것은 pivot이라 할 수 있다. 둘째, 변형 욕설 필터링 실험이다. 근사 단어 검색 시스템을 이용해 만든 변형 욕설 필터링 시스템의 성능을 측정하기 위하여 기존 시스템과 비교 실험을 수행한다.

5.1 실험 1 : pivot 갯수 및 길이 따른 검색 효율성 평가

우리는 앞서서 pivot의 수, 길이, 선택 방법에 따른 검색 성능 변화에 대하여 설명하였다. 본 장에서는 실제 실험을 통해 각 변인에 따른 성능에 변화에 대해서 분석하고자 한다. 먼저 pivot의 사용 개수에 따른 검색 효율성에 대해서 분석하겠다. 사용된 데이터베이스는 일반 국어사전 1000, 60000단어, 비속어 6000단어이다. 사용된 검색 반경은 3이며 각 수치 당 1000번 이상의 실험이 수행되었다. 검색반경은 초성, 중성, 종성을 합쳐서 완전한 1개의 글자가 변화되었을 때를 측정한 것이다.

표 14 Pivot 수에 따른 검색 효율성 실험

	일반단어 1000		비속어 6000	
BP수	시간	평균결과수	시간	평균결과수
10	0.0169	505.2	0.0716	2145.1
20	0.0110	331.3	0.0388	1214.2
30	0.0084	236.1	0.0244	762.5
40	0.0072	179.8	0.0176	541.1
50	0.0064	141.9	0.0152	441.9
60	0.0058	105.4	0.0132	356.5
70	0.0054	77.0	0.0114	287.9
80	0.0056	70.1	0.0109	262.1
90	0.0058	56.6	0.0102	211.0
100	0.0060	48.7	0.0104	182.5

본 시스템은 R*tree 검색을 통해 걸러진 후보군을 입력된 질의 단어와 전역 정렬 방식으로 유사도를 비교하여 최종적으로 가장 유사한 단어를 보이는 방식을 취한다. 즉 이 시스템에서 가장 큰 부분을 차지하는 것은 질의 단어와 pivot들과의 편집거리 계산과 후보군과 질의 단어 간의 유사도 계산이다. 즉 pivot수 + 평균결과수가 가장 적을 때 가장 높은 성능을 보인다고 할 수 있다. 위의 표 14에서 보면 일반단어 1000개가 데이터베이스인 경우 pivot 수 70, 비속어 6000개의 경우 90에서 최적화되는 것을 볼 수 있다. 즉 데이터베이스의 단어의 개수와 필요한 pivot의 숫자는 비례하지 않는 것을 알 수 있다. 메모리의 경우 비속어 6000개의 경우 4,448 Kbyte정도로 모바일 시스템에 적용하기에 큰 무리가 없는 것을 알 수 있다.

그림 6을 보면 pivot의 숫자에 따른 성능의 변화를 명확하게 알 수 있다. 오른쪽의 pivot수에 따른 평균결과단어수의 그래프를 보면 값은 pivot의 개수에 정비례하지 않고 점점 줄어드는 정도가 떨어지는 것을 알 수 있다. 이 사실은 pivot의 숫자를 정하는데 아주 중요한 요소이다. 즉 pivot의 증가는 평균 결과단어의 정량적인 감소가 아닌 비율적인 감소를 가져오게 된다.

그래프 7은 pivot숫자의 증가에 따른 pivot 10 증가당 평균 결과 단어의 감소 비율을 나타낸다. pivot의 수가 증가함에 따라 감소되는 비율역시 비례하여 감소하는 것을 알 수 있다. 이것은 pivot을 추가시키더라도 이미 기존의 pivot에 의해 분류된 단어의 일부를 다시 재분류하기 때문이다. 즉 pivot이 추가되더라도 데이터베이스의 단어를 균일하게 분배하는 것이 아니라 그 단어와 유사한 일부 영역에 대해서만 세부적인 분류가 이루어지고 나머지 단어에 대해서는 최대 편집거리로 계산되어 한 분류에 대부분의 단어가 집중되기 때문에 일어나는 현상이다. 즉 무조건적인 pivot 수의 증가가 시스

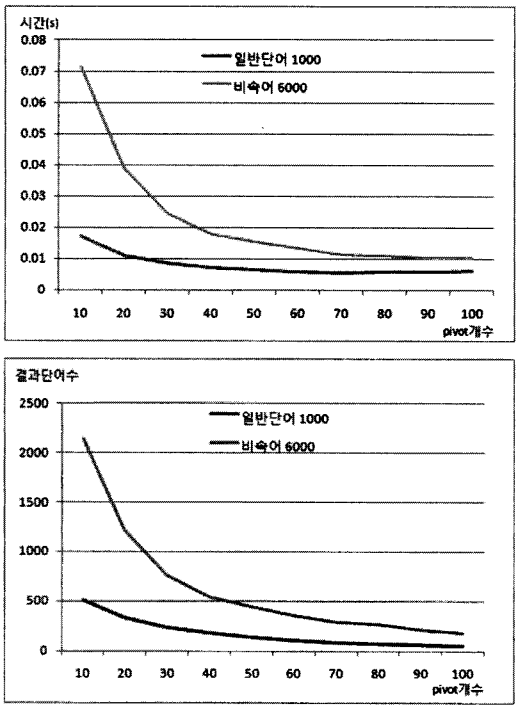

그림 6 사용되는 pivot의 수에 따른 성능변화 - pivot의 수가 증가함에 따라 평균 결과 단어 수가 감소하나 질의 단어와 pivot간의 편집거리 연산 시간이 증가하기 때문에 일정 수치를 기점으로 총 연산 효율의 증가가 멈추게 된다.

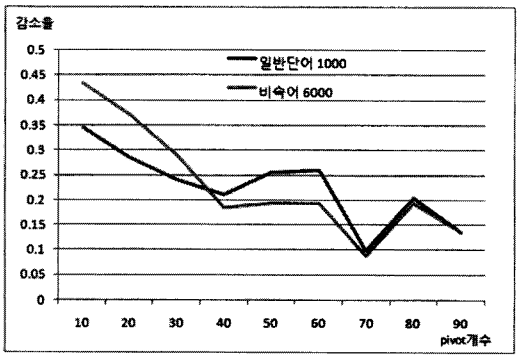

그림 7 pivot의 수 증가에 따른 pivot개수 10 증가당 결과단어 감소 비율 - pivot의 전체 수가 증가함에 따라 pivot 증가에 따른 결과숫자의 감소 비율이 전체적으로 낮아지는 것을 알 수 있다.

템의 성능향상에 도움을 주지 않는 다는 것을 보여준다.

그래프 8은 pivot의 평균 길이에 따른 검색 효율의 변화이다. 실험은 1000개의 일반단어에서 가장효율이 높은 80개의 피벗일때 이루어졌다. 검색 성능은 pivot의

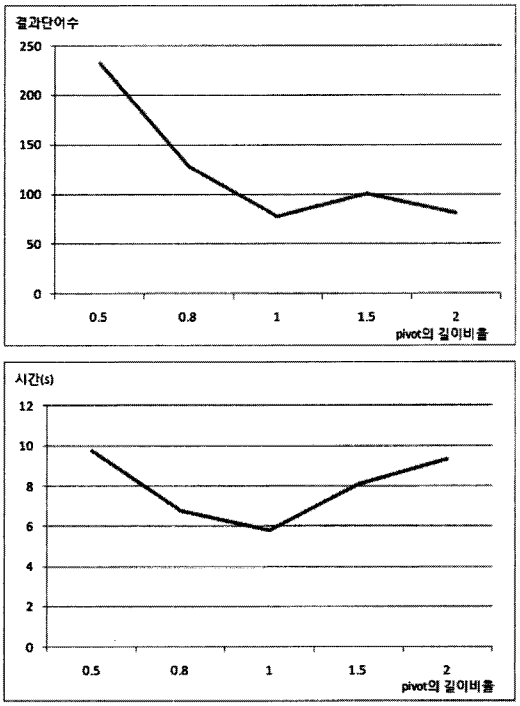

그림 8 pivot의 평균 길이에 따른 검색효율 - pivot의 평균길이가 데이터베이스내의 단어의 평균길이와 일치할 때 가장 높은 성능을 보이는 것을 알 수 있다.

평균 길이가 데이터베이스 단어들의 평균길이와 일치할 때 가장 성능이 높았다. pivot의 길이가 너무 짧을 경우 충분한 단어의 분류가 이루어지지 않고 pivot의 길이가 너무 길경우 두 가지 이상의 패턴을 가지고 있게 되어서 오히려 단어의 분류 효율성이 떨어지기 때문이다.

5.2 변형 욕설 필터링 실험

우리는 본 시스템의 성능을 측정하기 위하여 무작위로 수집된 일반 욕설 1,672개의 필터링 실험과 무작위로 추출된 정상 단어 1,505개에 대한 검증 실험을 수행하였다. 실험에 사용된 금칙어 리스트는 한국게임산업진흥원에서 배포한 "게임언어 건전화 지침서 연구"에 독자적으로 수집한 비속어를 추가하여 9,265개의 금칙어를 이용하여 실험하였다.

$$sensitivity = \frac{T_p}{T_p+F_n},\ specificity = \frac{T_n}{T_n+F_p} \quad (1)$$

위 공식은 성능의 기준으로 사용되는 Sensitivity와 Specificity의 공식이다. T_p(True Positive)는 필터링 된 단어들 중에서 욕설이 맞는 경우에 해당하고 F_n(False Negative)은 필터링 되지 않은 단어 중에서 욕설 수를 의미한다. T_n(True Negative)은 필터링 되지 않은 단어 중 정상 단어를 의미하고 F_p(False Positive)는 필터링 된 단어 중에서 정상 단어이다.

표 15 실험에 사용된 입력단어와 검출된 결과

	입력 단어 수	검출 단어수
정상 단어	1,505	18
일반 욕설	1,672	1,648

표 16 변형 욕설 필터링 비율 비교 실험 결과(200단어)

	검출된 단어 수	Sensitivity
제안된 시스템	180	90.0%
게임포탈(N사)	60	30.0%
게임포탈(H사)	23	11.5%
게임포탈(P사)	20	10.0%

(1)의 공식과 실험 결과(표 15)로 계산되는 Sensitivity와 Specificity는 각각 98.5%와 98.8%이다. Specificity가 아주 높게 측정되어 Scunthorpe problem이 상당 부분 해결된 것을 알 수 있다. 일반 욕설에 대한 Sensitivity 또한 문턱값(Threshold Value) 75%에서 거의 유사한 수치인 98.5%를 기록하여 일반적인 비속어 필터로서의 성능은 만족할만한 수준인 것을 알 수 있다.

표 16은 본 시스템과 유명 게임포탈 N사, H사와 P사의 금칙어 필터링 시스템 간의 변형 욕설 필터링 비율에 관한 비교 실험 결과이다. 입력된 단어는 일반 욕설을 의미가 통하는 정도로 변형 시킨 욕설들을 사용하였다. N사가 30% 정도, 다른 두 게임 포탈의 필터링 시스템이 각각 11.5%와 10%로 매우 낮은 성능을 보인 것에 반해 본 시스템에서는 문턱 값 75%를 적용하였을 때 90% 정도의 필터링 비율을 보여 단어 단위의 변형 욕설 필터링 문제에 대해 매우 우수한 성능을 보이는 것을 알 수 있다.

6. 결론 및 향후 연구 과제

우리는 모바일 기기 등의 제한된 입력도구와 연산 능력을 가지는 환경에서 효과적으로 입력 오류에 대처할 수 있는 근사 한글 단어 검색 시스템에 대하여 설명하였다. 본 시스템은 기존의 알파벳에 적용된 근사단어 검색 시스템을 한글에 효과적으로 적용하기 위하여 다음의 방법을 사용하였다.

1. 한글 단어를 위한 서열 정렬 알고리즘 : 자음과 모음의 조합으로 만들어지는 한글의 특성에 맞춰 한글을 자음과 모음으로 분리하여 발음에 가까운 단어 간에 유사도가 높게 나오게 조정하였다. 더불어 유사한 발음의 자소 간에 Matching Matrix를 구성하여 실제 입력 장치로 인한 오타 혹은 잘못된 기억으로 인한 입력 오류에 효율적으로 대처할 수 있게 하였다.

2. 근사 한글 단어 색인을 위한 대표형 변환 알고리즘 : 자음과 모음의 조합으로 지나치게 많은 글자의 조합이 가능하여 색인이 복잡해지는 것을 막기 위하여 자모를 분리, 유사한 발음의 글자를 통합하여 대표 문자로 변환하는 과정을 통해 만들어진 새로운 문자의 조합을 색인으로 사용하여 입력오류에 관계없이 단어의 검색을 가능하게 하였다.

3. 데이터베이스의 특성에 맞는 pivot 선택 알고리즘 : 데이터베이스의 특성에 따라 pivot의 수, 길이, 구성방법에 대하여 제안하고 실험을 통해 그 성능을 검증 하였다. 일반적인 국어사전의 크기인 60000단어에서 pivot 수 160에서 최적화된다는 사실과 비속어 사전 6000개에서 pivot수 90에서 최적화된다는 사실을 밝혀내었다.

4. 한글 근사 단어 검색 시스템을 적용한 변형 비속어 필터링 시스템 : 기존의 비속어 필터링 시스템은 정해진 단어만 필터링 할 수 있었다. 우리는 제안된 한글 근사 단어 검색 시스템을 적용하여 사용자가 데이터베이스에 존재하지 않는 변형 비속어를 입력하더라도 해당 비속어를 데이터 베이스 내의 단어와 비교하여 필터링을 수행할 수 있는 비속어 필터링 시스템을 제안하였다.

본 시스템의 한글 근사 단어 검색 시스템은 충분히 적용 가능한 속도와 정확도를 실험을 통해 증명하였다. 그리고 변형 비속어 필터링 시스템 역시 충분한 Sensitivity와 Specificity를 보여 90% 이상의 변형 비속어를 필터링하여 기존의 필터링 시스템이 가진 문제를 상당 부분 해결하였다.

본 시스템의 향후 연구 방향으로 가장 중요한 점은 pivot의 선택 방법을 개량하는 것이다. 검색의 성능에 가장 큰 영향을 주는 요소라 할 수 있는 pivot선택 알고리즘을 더 많은 실험을 통하여 정확한 관계식을 구하는 일이다. 그리고 모바일 기기의 입력 장치에 대한 연구를 추가하여 입력 장치에 특화된 대표형 변환 알고리즘을 개발하는 것이다. 입력 위치가 가까운 글자 간의 관계성을 추가하여 기기로 인한 입력오타에 좀 더 효율적으로 대처할 수 있게 만드는 것이다.

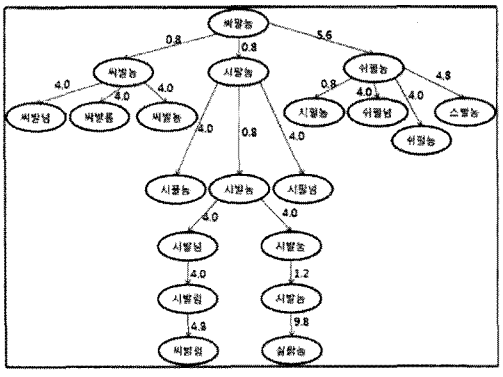

그림 9 욕설 단어의 진화 계통도 분석

근사 한글 단어 검색 시스템을 이용한 변형 비속어 필터링의 시스템의 경우 비속어 단어 간의 유사도를 이용하여 비속어의 발전 과정을 추적하여 그림 9와 같이 단어의 진화에 따른 변화 양상을 파악하여 사용자가 입력할 변형 비속어를 미리 예측하여 효과적인 비속어 필터링이 가능하도록 데이터베이스를 구성하도록 하는 것이다.

참 고 문 헌

[1] A. Apostolico. The myriad virtues of subword trees. Combinatorial Algorithms on Words, pp.85-96, 1985.

[2] W. A. Burkhard and R. M. Keller. Some approaches to best-match file searching. Commun. ACM, vol.16, no.4, pp.230-236, 1973.

[3] Chang-Keon Ryu, Hyong-Jun Kim, Seung-Hyun Ji, Gyun Woo, and Hwan-Gue Cho. Detecting and tracing plagiarized documents by reconstruction plagiarism-evolution tree. Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on, pp.119-124, July 2008.

[4] Hyong-Jun Kim, Chang-Keon Ryu, and Hwan-Gue Cho. A detecting and tracing algorithm for unauthorized internet-news plagiarism using spatio-temporal document evolution model. In SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing, pp.863-868, New York, NY, USA, 2009. ACM.

[5] Chang-Keon Ryu, Hyong-Jun Kim, and Hwan-Gue Cho. Reconstructing evolution process of documents in spatio-temporal analysis. In ICCIT '08: Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology, pp.136-142, Washington, DC, USA, 2008. IEEE Computer Society.

[6] Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, and D.J. Basic local alignment search tool. Journal of Molecular Biology., 215, 1990.

[7] K. M. Chao and L. Zhang, Sequence Comparison Theory and Methods, Springer, 2009.

[8] Sreenivas Gollapudi and Rina Panigrahy. A dictionary for approximate string search and longest prefix search. In CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management, pp.768-775, New York, NY, USA, 2006. ACM.

[9] Trinh N. D. Huynh, Wing-Kai Hon, Tak-Wah Lam, and Wing-Kin Sung. Approximate string matching using compressed suffix arrays. Theoretical Computer Science, vol.352, no.1, pp.240-249, 2006.

[10] Marios Hadjieleftheriou, Nick Koudas, and Divesh Srivastava. Incremental maintenance of length normalized indexes for approximate string matching. In SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data, pp.429-440, New York, NY, USA, 2009. ACM.

[11] Gonzalo Navarro and Edgar Chavez. A metric index for approximate string matching. Theoretical Computer Science, vol.352, no.1, pp.266-279, 2006.

[12] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In SIGMOD '90: Proceedings of the 1990 ACM SIGMOD international conference on Management of data, pp.322-331, New York, NY, USA, 1990. ACM.

[13] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. Readings in database systems, pp.599-609, 1988.

[14] http://en.wikipedia.org/wiki/swearfilter.

[15] Korea Game Industry Agency, Sound Game language guide research, 2008.

[16] Shekhar Dhupelia. Designing a vulgarity filtering system. In Game Programming Gems 5. Charles River Media, 2005.

[17] Lai C. An empirical study of three machine learning methods for spam filtering. Know-Based System, vol.20, no.3, pp.249-254, 2007.

[18] Kyo Hyeon Park and Jee Hyong Lee. Developing a vulgarity filtering system for online games using svm. In Proceedings of the Korean Institute of Information Scientists and Engineers Autumn, 2006.

[19] Ramachandran A Feamster N and Vempala S. Filtering spam with behavioral blacklisting. In Proceedings of the 14th ACM Conference on Computer and Communications Security (Alexandria, Virginia), pp.342-351, 2001.

[20] Imoxion, Lewdness/Profanity Filtering System Using Syllable Information, patent, In 2001-0067853, 2001.

[21] http://en.wikipedia.org/wiki/scunthorpe problem.

윤 태 진

2009년 부산대학교 정보컴퓨터공학부 졸업(학사) 2009년~현재 부산대학교 정보컴퓨터공학부 석사과정. 관심분야는 한글 언어처리, 정보 검색, 이미지 프로세싱

조 환 규

1984년 서울대학교 계산통계과 졸업(학사). 1990년 KAIST 대학원 전산학과(공학석사, 공학박사). 1990년~현재 부산대학교 정보컴퓨터공학부 교수. 1994년~현재 한국정보올림피아드 운영위원. 2007년~2008년 정보과학회 컴퓨터이론연구회 연구위원장. 관심분야는 알고리즘, 응용 그래프 이론, 생물정보학

정 우 근

2009년 부산외국어대학교 정보컴퓨터공학부 졸업(학사). 2009년~현재 부산대학교 정보컴퓨터공학부 석사과정. 관심분야는 생물정보학, CAPTCHA, 이미지 프로세싱