# 효율적인 근사 단어 검색을 위한 pivot 선택 방법론

Pivot Choosing Method for Efficient Approximate String Search

윤태진
Yoon Taijin
부산대학교 컴퓨터공학과
ytj@pusan.ac.kr

ABSTRACT

한글 근사 단어 검색은 변형 비속어 필터링의 속도 문제를 해결하기 위한 핵심적인 요소라고 할 수 있다. 정확한 검색이 이루어질 수록 적은 수의 후보단어가 추출되어 유사도 측정에 필요한 서열 정렬의 횟수를 줄여 줄 수 있기 때문이다. 우리의 비속어 필터링 시스템에서는 근사단어 검색을 위하여 Metric Space의 성질을 이용한 다차원 자료구조를 사용하는데 이 자료구조에서 좌표축이 되는 pivot의 추출 방법은 시스템의 효율을 결정하는데 아주 중요한 요소이다. 데이터베이스의 단어를 가능한 균일하게 분포시키는 것이 시스템의 효율을 증가시키는 것과 연결 되기 때문이다. 이 보고서에서는 기존 논문에서의 pivot 추출 방법과 한글 근사 단어 검색 시스템을 위한 추가적인 아이디어에 대하여 설명하도록 하겠다.

KEYWORDS Approximate string matching, Metric spaces, Information Retrieval

## 1 서 론

우리는 변형 비속어를 필터링 하기 위하여 서열 정렬 알고리즘을 이용하여 데이터베이스의 단어와 입력 단어 간의 유사도를 측정하여 기준치 이상의 단어를 필터링 하는 방식을 사용한다. 그러나 데이터베이스의 모든 단어와 입력 단어의 유사도를 측정하는 방식은 너무 비효율적이므로 근사 단어 검색 시스템을 이용하여 입력단어와 유사한 단어를 추출하여 추출된 단어만 검사하는 방식을 사용한다.

이 근사 단어 검색 시스템으로 우리는 편집 거리가 Metric Space를 만족하는 성질을 이용하여 구성된 다차원 자료구조의 범위 검색 방법을 사용하게 된다. 이 자료구조는 데이터베이스에서 추출된 단어인 pivot과 데이터베이스의 단어 사이의 편집거리가 각 차원의 좌표가 되므로 이 pivot을 어떻게 구성하는가에 따라 검색의 정확성이 크게 변화된다. 예를 들어 "가"와 같은 너무 짧은 단어를 pivot으로 선택할 경우 "ㄱ"과 "ㅏ"가 포함된 단어를 제외한 모든 단어가 한 구역으로 뭉치게 되므로 효율성이 떨어지고 반대로 너무 긴 단어를 pivot으로 선택할 경우 모든 단어에 대해 유사한 부분이 존재하게 되어 오히려 특정 위치에 단어를 집중시켜 분포 효율을 떨어뜨리게 된다. 이러한 길이 외에도 추출된 단어의 다양성 등 여러가지 요소가 필터링 효율을 결정하는 기준이 된다.

기존의 근사 단어 검색에서 사용하는 방법은 영문 알파벳이나 DNA 서열과 같은 한정된 알파벳에서 유용한 pivot 선택 방법을 사용하고 있다. 본 보고서에서는 기존의 pivot 선택 방법을 분석하고

이를 한글 근사 단어 검색 시스템에 맞는 추가적인 아이디어에 대하여 설명하도록 하겠다.

## 2 기존의 근사 단어 검색 시스템의 **pivot** 선택 방법

아무 pivot 선택 방법을 사용한다고 하여 좋은 성능을 보장할 수는 없다. 특히 무작위로 $k$ 개의 pivot 을 선택할 경우 제대로된 결과를 보여주지 못한다. 길이 $n$ 의 문자열 $T$, 길이 $m$ 의 패턴 $P$, 최대의 편집거리 $r$, 알파벳의 사이즈 $\sigma$ 라 할때 평균적인 시간 복잡도로 $\sigma^r$ 를 보인다.

우수한 성능을 보이는 전략은 다음과 같다. 어떤 작은 양의 정수 $q$(나중에 유도된다.)를 선택한다. $q$ 의 배수가 되도록 $T$ 에 무작위 문자를 덧붙인다. $T$ 의 $q$-gram 인 새 문자열 $T'$ 를 만든다. $T' = T_{1...q}T_{q+1...2q}...T_{n-q+1...n}$($T'$ 의 각 캐릭터는 $T$ 의 $q$-gram). 이것은 길이가 $\sigma' = \sigma^q$ 인 $\sum^q$ 의 길이가 $n/q$ 의 올림인 문자열과 동등하다. $P_{1...m}$ 을 찾기 위해서 우리는 $P$ 를 길이 $m_t, m-q < m_t \leq m$ 으로 자르고, $m_t + 1$ 을 $q$ 로 나누고, 패턴 $P^s = P_{s+1...s+q}P_{s+q+1...s+2q}...P_{s+m_t-2q+2...s+m_t-q+1}$ , $0 \leq s < q$ 를 위한 q 검색을 수행한다. 모든 $P^s$ 는 $m' = (m_t+1)/q - 1 > m/q - 2$ 의 길이이다($q$-grams 에서 측정되었다). $q$ 검색을 수행하는 것은 $T'$ 의 $q$-grams 안의 $P$ 에서 일어나는 모든 가능한 다른 정렬을 설명하기 위해 필요하다. 우리는 같은 문턱값 $r$ 을 보수하여야한다. 만약 $P$ 가 $T_{i...j}$ 에서 $d$ 의 거리를 가지면, $q$-gram 들은 최대 $d$ 거리 안에 있다(각 오류는 다른 $q$-gram 으로 수정된다.).

문자열 $T'$ 에서, 우리는 각 $c_i \in \sum^q$ 에 대하여 $p_i = c_i^{m'}$ 의 형태의 $k = \sigma'$ pivot 을 선택한다($m'q$-grams $c_i$ 의 서열). 이 경우, $ed(P^s, p_i)$ 는 $P^s$ 에서 $c_i$ 의 발생수의 $m'$ 마이너스이다($c_i$ 는 길이 $q$ 의 문자열이 아니라 하나의 문자로 취급한다). 모든 $P^s$ 로의 모든 거리들은 $O(q\sigma^q + m)$ 시간 안에 계산된다: 우리는 $ed(P^s, p_i) = m'$ 에서 모든 $q\sigma^q$ 거리를 초기화하고 $P$ 의 모든 $m_t - q + 1q$-grams 를 한번씩 방문한다. 그래서 스텝 $j$ 에서 우리는 $ed(P^s, p_i)$ 거리를 1 감소시킨다. $P_{j...j+q-1} = c_i$ 이고 $s = jmodq$ 이다. 이것은 우리의 일반적인 분석과 다르다. 우리의 pivot 들의 특정 구조의 얻는 이득에 대해 $O(klm)$ 보다 적게 지불한다.

$$ed(P, p_i) + r < min(ed(x[y], p_i)) \bigvee ed(p, p_i) - r > max(ed(x[y], p_i)) \quad (1)$$

$P^s$ 에 잠시 주목 하자. 비관적으로 덧붙이자면 $O(n')$ 의 suffix tree node 인 $x[y]$ 에 대하여, 우리는 Eq.(1) 의 더 느슨한 조건을 대신하여, $ed(P^s, p_i) + r < mined([xy], p_i)$ 일때만 제거한다. 그들의 복잡도는 $n'$ 의 문자열 suffix $[T'_{j...}]$ 를 인덱싱하는 것과 동등하다. 우리의 pivot 들에 대해서 $mined([T'_{j...}], p_i) \geq ed(T'_{j...j+m'-1}, p_i)$ 를 아는 것은 쉬운일이다. 하지만 우리는 $ed(P^s, p_i) < ed(T'_{j...j+m'-1}, p_i)$ 일때마다 $j$ 위치의 문자열을 버린다.

우리가 모든 $\sigma'$ pivot 들을 버리는데 사용한다고 하면, $j$ 위치의 문자열은 제거된다, 만약 어떤 $q$-gram $c_i$ 에 대하여, $P^s$ 안의 $c_i$ 의 발생수 더하기 $r$ 은 여전히 $T'_{j...j+m'-1}$ 안의 $c_i$ 발생수에 미치지 못

한다. 주어진 패턴 $P^s$ 에 대하여 버려질 수 없는 다른 문자열들의 집합은 모든 가능한 방법으로 바꾼 $q$-gram 들에 포함되어 있으며, $r$ 위치들을 선택하고, 그 위치에서 $q$-gram 들을 변경시킨다(그들 스스로 가능한 방법으로). 그 방법으로 가능한 최대 문자열을 숫자는 $m'!\binom{m'}{r}(\sigma')^r = O((m')^{m'+r+1/2}(\sigma')^r/e^{m'})$ 이다.

suffix $[T'_{j\ldots}]$ 가 버려지지 않을 확률은 정확하게 $T'_{j\ldots j+m'-1}$ 의 집합을 따라서 최대

$$p = p(m', r) = O\Big(\frac{(m')^{m'+r+1/2}(\sigma')^r}{e^{m'}(\sigma')^{m'}}\Big) = O\Big(\frac{(m/q)^{m/q+r+1/2}\sigma^{qr}}{e^{m/q}\sigma^{m-2q}}\Big)$$

우리가 $P^s$ 의 다른 패턴들을 위해 $q$ 검색을 수행하는 것을 상기해보자. 검색 비용은

$$O(q\sigma^q + m + qn'p(m', r)(m^2 + \log n')), \quad (2)$$

$qn' = n$ 이다. $p(m', r) = O(1/(nm\log n))$. logarithm 을 취하면 우리는 다음을 얻는다.

$$\Big(\tfrac{m}{q} + r + \tfrac{1}{2}\Big)log_\sigma \tfrac{m}{q} + qr - \tfrac{m}{q}\log_\sigma e - m + 2q \le -\log_\sigma n - \log_\sigma m - \log_\sigma \log n,$$

혹은 더 엄격하게 한다면,

$$m \ge \log_\sigma n + \log_\sigma \log n + q(r+2) + \Big(\tfrac{m}{q} + r + \tfrac{3}{2}\Big)\log_\sigma m,$$

우리는 충분히 큰 m 에 대하여 이 조건이 충족된다는 것에 주목한다.

만약 $(m/q)\log_\sigma m \ge m$ 이면 조건을 만족하지 않는다. 그러므로, 상수 $\alpha < 1$ 에 대하여 $q \ge (1/\alpha)\log_\sigma m$ 이 필요하다. 유사하게, 우리는 상수 $\beta < 1$ 에 대하여 $q \le \beta m/(r+2)$ 를 필요로 해서 $q(r+2)$ 는 $m$ 의 한계에서 벗어난다. 이 $q$ 에 대한 조건은 $r + 2 \le \alpha\beta m/\log_\sigma m$ 로 나타난다. 만약 덧붙여서, $\alpha + \beta + \alpha\beta < 1$ 이면 조건은 다음을 만족한다.

$$m \le \frac{\log_\sigma n + \log_\sigma \log n}{1 - \alpha - \beta - \alpha\beta}.$$

$\alpha + \beta + \alpha\beta < 1$ 이라는 조건과 함께 $r$ 의 제한을 생각하면, $r + 2 < (3 - 2\sqrt{2})m/log_\sigma m(3 - 2\sqrt{2} \approx 0.172)$. 주어진 $r$ 이 이 조건을 만족하면, 우리는 언제든지 알맞은 $\alpha$ 와 $\beta$ 의 값을 찾을 수 있고 적당한 $q$ 는 다음 사이에 있다$[(1/\alpha)\log_\sigma m, \beta m/(r+2)]$.

그러므로, 우리는 Eq. (2) 가 $O(m)$ 이 되는 세번째 항의 조건을 찾는다. 첫번째 항은 $q\sigma^q$ 이다. 그러므로 우리는 $q$ 가 가능한 작기를 바라며, $q = (1/\alpha)\log_\sigma m$, 이므로 $q\sigma^q = O(m^{1/\alpha}\log_\sigma m)$ 이다. 이 항은 복잡도를 좌우하고, 우리는 $\alpha$ 가 최대한 1 에 가깝기를 원한다. 이 최대한 가능한 값은 $\gamma =$

$(r+2)\log_\sigma(m)/m$ 일때 $\alpha < (1-\gamma+\sqrt{\gamma^2-6\gamma+1})/2$ 이다. 그러므로 모든 상수 $\varepsilon > 0$ 에 대하여 복잡도는 $O(m^{1/\alpha+\varepsilon}) = O(m^{2/(1-\gamma+\sqrt{\gamma^2-6\gamma+1})+\varepsilon})$ 이다. 이것은 만약 $r = o(m/\log_\sigma m)$ 이면 $O(m^{1+\varepsilon})$ 이다 ($\gamma = o(1)$). $\gamma$ 가 $\gamma < 3-2\sqrt{2}$, 때까지 증가함에 따라, 우리는 $\alpha$ 의 한계를 낮춘다, 최대 $\alpha < \sqrt{2}-1$ 까지. 따라서, 최고의 오류 수준을 위하여 우리가 가질 수 있는 최대복잡도는 $O(m^{1+\sqrt{2}+\varepsilon})$ 이다($1+\sqrt{2} \approx 2.4142$).

$m$ 의 사용가능한 조건은 우리가 원하는 복잡도에 의존한다. 만약 $r = o(m/\log_\sigma m)$ 이면 우리는 $O(m^{1/\alpha+\varepsilon})$ 을 $m > \log_\sigma n/(1-\alpha)$ 에 대하여 얻을 수 있으며, 그것은 $m > ((1+\varepsilon)/\varepsilon)\log_\sigma n$ 에 대하여 $O(m^{1+\varepsilon})$ 이다. 상수 $\gamma$ 에 대하여, $m > \log_\sigma n/(1-\alpha-(1+1/\alpha)\gamma)$ 로 제한된다.

구성에 필요한 시간과 공간을 고려해 보자. 한 suffix tree 에 대하여 모든 pivot 과의 거리를 동시에 계산하는 것은 그렇게 어렵지 않으며, 루트에서 현재 노드까지 패스에서 몇번 발생하는지 셀 뿐이다. 그러므로 이 부분에서는 단지 평균적으로 $O(\sigma^q+n)$ 의 시간이 걸리고 최악의 경우 $O(\sigma^q+n^2)$ 의 시간이 걸린다.

우리의 접근의 문제점은 우리는 $q$ 를 선택하는 인덱싱때 반드시 $m$ 과 $r$ 을 알아야 한다. 다른 값을 적용하더라도 올바르게 동작할 수는 있으나 여기에서 보여진 복잡도와는 차이가 있을것이다. 많은 시스템에서 $m$ 과 $r$ 이 알려져 있기에 문제는 없으나 다른 종류의 시스템에서는 대체할 방법이 필요하다.

## 3 비속어 필터링 시스템을 위한 **pivot** 선택 방법

앞에서 설명한 pivot 선택 방법은 일정 길이의 문자열 내에서 pivot 을 추출하는 방법으로 우리가 목표로 하는 단어의 리스트를 이용한 근사 단어 검색 시스템에 그대로 적용하기에는 무리가 있다. 특히 알파벳과 한글의 차이에서 오는 문제점을 해결하기 위해서는 한글의 특성에 맞는 개량이 필요하다.

먼저 한글의 특성은 초성, 중성, 종성의 조합을 통해 이루어지게 되어 있으므로 단순히 편집거리를 측정할 경우 매우 유사한 단어도 서로 아주 먼 거리로 측정되게 되어 있다. 예를 들어 "청소"와 "처송"은 매우 유사한 단어지만 편집거리가 2로 단어의 길이에 비해 상대적으로 매우 멀게 측정된다. 단어의 길이 역시 문제가 되는데 한글은 대부분 4자 이하의 단어로 이루어져 있어 편집거리를 좌표로 사용하려고 해도 매우 좁은 영역에서 단어의 분류가 이루어지게 된다. 이러한 문제점을 해결하기 위하여 우리는 단어의 자모를 분리시켜 좌표공간을 더 넓게 사용하고 사람이 느끼기에 더 납득할 수 있는 단어의 분포를 만들 수 있다.

pivot 으로 선택되는 단어의 길이 역시 중요한 요소이다. 앞서 말했듯이 지나치게 짧은 단어는 제대로된 단어의 분류가 이루어지지 못하게 된다. 길이가 지나치게 긴 단어 역시 마찬가지인데 길이가 길다는 것은 내부에 여러가지 패턴을 동시에 포함하게 된다는 것이다. 예를 들어 "가나다라마바"라는

단어는 "가나다"라는 단어와 "라마바"라는 단어를 같은 편집거리로 측정하여 전혀 다른 두 단어를 같은 위치에 분포시키게 된다. 그리고 pivot의 길이를 일정하게 통일 시키는 것 역시 좋지 않은데 이것은 단어를 고르게 분포시키는데 필요한 패턴의 다양성을 해치게 되기 때문이다.

pivot을 고르는데 가장 중요한 요소는 패턴의 다양성이라 할 수 있다. pivot을 수십, 수백개 사용하여 단어를 분류하더라도 모두 같은 단어라면 의미가 없기 때문이다. 그러나 이 다양한 패턴의 단어를 추룰하는 일은 쉬운일이 아니다. 이미 추출된 단어와 가장 편집거리가 먼 단어를 추출할 경우 필연적으로 길이가 먼 단어가 우선시해서 뽑힌다거나 일반적으로 잘 쓰이지 않는 유형의 단어만 추출되어 많이 쓰이는 패턴의 단어들은 분류하는데 문제가 생기기 때문이다. pivot의 특성상 유사한 단어를 더 세밀하게 분류하게 되는데 너무 특이한 패턴의 단어들이 pviot으로 선정되면 이러한 많은 영역을 차지하는 단어들을 분류하는데 어려움이 따르게 된다.

위의 문제는 단어를 필터링하는데 어떠한 방법론을 추구하는데 걸림돌이 된다. 규칙의 추가가 단어를 필터링하는 효율성을 오히려 떨어뜨리게 되기 때문이다. 실제로도 실험에서 pviot 선택에 규칙을 추가할 경우 오히려 효율을 떨어뜨리는 경우가 많았다. 그러나 단순한 무작위 추출 방법으로는 데이터베이스 단어 수의 증가에 따른 pivot의 증가와 결과의 증가에 따른 총 연산시간의 증가로 인해 수만개 규모 이상의 대규모 데이터 베이스의 경우 실제 사용하는데 문제가 생기게 된다.

즉 좋은 pivot이란 기존의 pivot과 거리가 먼 pivot이 아니라 기존의 pivot들이 가지지 않은 패턴을 포함하고 잇는 단어를 의미한다. 이를 위해서는 단어 전체를 기존의 단어와 비교하는 것이 아닌 suffix tree를 통해 단어를 분해하여 패턴을 추출하여 이 추출된 패턴을 기존의 pivot들과 비교하여 중복되지 않는 패턴 즉 패턴의 총길이에 해당하는 매칭 값을 가지지 않는 다면 이것을 pivot에 추가하는 방법을 사용하게 된다. 이를 위하여 편집거리가 아닌 편집거리의 역, 즉 매칭되는 패턴의 길이를 측정하여 모든 단어와의 이 수치 중에 패턴의 길이에 해당하는 수치가 없을 경우 이것을 pivot으로 선택하게 된다. 이 경우 지역 정렬 방법을 사용하여 효율성을 높일 수 있다.

## 4 결론

본 보고서는 비속어 필터링 시스템의 근사 단어 검색시스템의 효율성을 높이기 위하여 효과적인 pivot을 선택하는 방법에 대하여 서술하였다. 기존의 근사 단어 검색 방법의 경우 일정한 크기의 문자열에서 pivot을 추출하기 위한 방법이므로 그것을 그대로 적용하기에는 적합하지 못하였다. 그래서 본 보고서에서는 한글 변형 비속어 데이터베이스에 적합한 pivot의 선정을 위하여 pivot의 길이와 패턴의 다양성 측면에서 접근하여 suffix tree를 이용한 해결방법을 제시하였다.

추후에는 시스템의 실제 구현과 실험을 통하여 본 보고서에서 제시된 방법론의 검즐을 수행하고 데이터를 바탕으로 수학적 검증을 수행하여야 할 것이다. 더불어 다단계 pivot 구조와 규합을 통하여

높은 효율성을 가진 실제 적용가능한 변형 비속어 필터링 시스템을 구현하도록 하겠다.

## 참고 문헌

1. Gonzalo Navarroa and Edgar Chávez, "A metric index for approximate string matching," *Theoretical Computer Science*, vol. 352, no. 1, pp. 266–279, 2006.
2. A. Apostolico, "The myriad virtues of subword trees," *Combinatorial Algorithms on Words*, pp. 85–96, 1985.
3. D. Gusfield, "Algorithms on strings trees and sequences," *Computer Science and Computational Biology*, 1997.
4. U. Manber and E. Myers, "Suffix arrays: a new method for on-line string searches," *SIAM J. Comput.*, p. 935–948, 1993.