n-gram/2L: 공간 및 시간 효율적인 2단계 n-gram 역색인 구조

(n-Gram/2L: A Space and Time Efficient Two-Level n-Gram Inverted Index Structure)

김 민 수 † 황 규 영 †† 이 재 길 ††† 이 민 재 ††††
(Min-Soo Kim) (Kyu-Young Whang) (Jae-Gil Lee) (Min-Jae Lee)

요 약 n-gram 기반 역색인 구조는 언어 중립적이고 에러 허용적인 장점들로 인해 일부 아시아권 언어에 대한 정보 검색이나 단백질과 DNA의 sequence의 근사 문자열 매칭에 유용하게 사용되고 있다. 그러나, n-gram 기반의 역색인 구조는 색인의 크기가 크고 질의 처리 시간이 오래 걸린다는 단점들을 가지고 있다. 이에 본 논문에서는 n-gram 기반 역색인의 장점을 그대로 유지하면서 색인의 크기를 줄이고 질의 처리 성능을 향상시킨 2단계 n-gram 역색인(간단히 n-gram/2L 역색인이라 부른다)을 제안한다. n-gram/2L 역색인은 n-gram 기반 역색인에 존재하던 위치 정보의 중복을 제거한다. 이를 위해 문서로부터 길이 m의 m-subsequence들을 추출하고, 그 m-subsequence들로부터 n-gram을 추출하여 2단계로 역색인을 구성한다. 이러한 2단계 구성 방법은 이론적으로 의미 있는 다치 종속성이 존재하는 릴레이션을 정규화하여 중복을 제거하는 것과 동일하며, 이를 본문에서 정형적으로 증명한다. n-gram/2L 역색인은 데이타의 크기가 커질 수록 n-gram 역색인에 비해 색인 크기가 줄어들며 질의 처리 성능이 향상되고, 질의 문자열의 길이가 길어져도 질의 처리 시간이 거의 증가하지 않는 좋은 특성을 가진다. 1GByte 크기의 데이타에 대한 실험을 통하여, n-gram/2L 역색인은 n-gram 기반 역색인에 비해 최대 1.9 ~ 2.7배 더 작은 크기를 가지면서, 동시에 질의 처리 성능은 3~18 범위의 길이를 가지는 질의들에 대해 최대 13.1배 향상됨을 보였다.

키워드 : 정보검색, 역색인, n-gram, 다치 종속성

Abstract The n-gram inverted index has two major advantages: language-neutral and error-tolerant. Due to these advantages, it has been widely used in information retrieval or in similar sequence matching for DNA and protein databases. Nevertheless, the *n-gram inverted index* also has drawbacks: the size tends to be very large, and the performance of queries tends to be bad. In this paper, we propose the *two-level n-gram inverted index* (simply, the *n-gram/2L index*) that significantly reduces the size and improves the query performance while preserving the advantages of the n-gram inverted index. The proposed index eliminates the redundancy of the position information that exists in the n-gram inverted index. The proposed index is constructed in two steps: 1) extracting subsequences of length m from documents and 2) extracting n-grams from those subsequences. We formally prove that this two-step construction is identical to the relational normalization process that removes the redundancy caused by a non-trivial multivalued dependency. The n-gram/2L index has excellent properties: 1) it significantly reduces the size and improves the performance compared with the n-gram inverted index with these improvements becoming more marked as the database size gets larger; 2) the query processing time increases only very slightly as the query length gets longer. Experimental results using databases of 1 GBytes show that the size of the n-gram/2L index is

· 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단으로부터 지원을 받았음

† 학생회원 : 한국과학기술원 전산학과
mskim@mozart.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
kywhang@mozart.kaist.ac.kr
††† 정 회 원 : 한국과학기술원 전산학과
jglee@mozart.kaist.ac.kr
†††† 정 회 원 : (주)네오위즈 연구소 연구원
Minjae.lee@gmail.com

논문접수 : 2005년 6월 15일
심사완료 : 2005년 10월 1일

reduced by up to 1.9 ~ 2.7 times and, at the same time, the query performance is improved by up to 13.1 times compared with those of the n-gram inverted index.

Key words : Information Retrieval, Inverted Index, n-gram, Multivalued Dependency

1. 서 론

텍스트 문서 데이타에 대한 검색은 매우 기본적이고 중요한 연산으로서 일상 생활에서 널리 사용되고 있다. 그 예는 정보 검색[1], 단백질과 DNA의 sequence에 대한 유사 sequence 매칭[2] 등이 있다. 단백질과 DNA의 sequence는 특별한 알파벳(예를 들어 DNA의 경우에는 {A,C,G,T})에 대한 텍스트 문서 데이타로 간주된다[3]. 이러한 텍스트 문서 데이타에 대한 검색 연산을 효율적으로 처리하기 위해 다양한 색인 구조들이 연구되었으며, 그 중 실용적으로 가장 널리 사용되는 색인 구조는 역색인(inverted index)[1]이다.

역색인은 색인에 사용되는 용어의 종류에 따라 단어를 색인 용어로 사용한 **단어 기반의 역색인**과 n-gram을 색인 용어로 사용한 **n-gram 기반의 역색인**(간단히 **n-gram 역색인**이라 부른다)으로 구분된다[4]. 단어는 문서로부터 언어적인 의미를 가지는 단위로서 추출되는 문자열이고, n-gram은 문서로부터 기계적인 방식으로 추출되는 길이 n의 문자열이다. 일반적으로 n-gram들은 문서상에서 길이 n의 문자열 윈도우를 1씩 슬라이딩 시키는 방식(간단히 **1-슬라이딩 방식**이라 부른다)으로 추출되기 때문에, 같은 문서로부터 추출된 n-gram들의 개수가 단어들의 개수보다 훨씬 더 많다.

n-gram 역색인은 언어 중립적(language-neutral)이고 에러 허용적(error tolerant)인 장점들을 가진다[1,5,6]. 언어 중립적이란 색인 용어를 기계적인 방식으로 추출하므로 언어적인 지식을 필요로 하지 않는다는 것을 의미한다. 이러한 특성으로 인해 n-gram 역색인은 복잡한 언어적인 지식을 필요로 하는 일부 아시아권 언어에 대한 정보 검색이나 단어의 개념이 명확하지 않은 단백질과 DNA의 sequence에 대한 검색에 많이 사용된다. 에러 허용적이란 색인 용어를 1-슬라이딩 방식으로 추출하므로 어떤 문서에 약간의 철자 에러가 있어도 그 문서를 검색할 수 있다는 것을 의미한다. 이러한 특성으로 인해 n-gram 역색인은 근사 문자열 매칭처럼 에러를 허용하여 문서를 검색하는 응용에 유용하게 사용된다.

그러나, n-gram 역색인은 색인의 크기가 크고 질의 처리 시간이 오래 걸리는 등 공간적, 시간적 비용이 크다는 단점을 가진다[1,5]. 이는 n-gram을 1-슬라이딩 방식으로 추출하는 용어 추출 방법의 특성 때문이다. 1-슬라이딩 방식으로 추출하므로 매우 많은 수의 n-gram들이 추출되고, 따라서 색인의 크기가 커진다. 또한, 질의 처리시 읽어 들여야 하는 포스팅들의 개수가 많아져 질의 처리 시간도 오래 걸린다.

n-gram 역색인의 크기가 큰 문제를 해결하기 위해 많은 연구들이 있었다. 가장 대표적인 연구로는 역색인 압축 기법[4,7,8]이 있다. 역색인 압축 기법은 색인의 구조는 그대로 유지하면서, 색인 구성시 포스팅 리스트를 압축하여 저장하고, 질의 처리시 포스팅 리스트를 읽어 들여 복원한다. 이로 인해 추가적인 압축, 복원 비용이 소요된다. 그 밖에, 색인의 크기를 줄이기 위한 연구로는 문서로부터 우선 단어를 추출한 후 단어 내에서만 n-gram들을 추출하거나[9], 단어 내에서 하나의 n-gram만을 추출함으로써[5] 색인할 용어의 개수를 줄이는 방법들이 있었다. 그러나, 이 방법들은 단어를 추출하기 어렵거나 단어의 개념이 없는 문서 데이타에는 적용하기 어렵다는 문제가 있다.

본 논문에서는 n-gram 역색인의 장점을 그대로 유지하면서 색인의 크기를 줄이는 **2단계 n-gram 역색인**(간단히 **n-gram/2L 역색인**이라 부른다)을 제안한다. 우선 n-gram 역색인의 크기가 커지는 원인을 분석하고, 그 주요 원인이 n-gram의 위치 정보의 중복 때문임을 밝힌다. n-gram의 **위치 정보**란 n-gram이 어떤 문서의 어떤 오프셋에 나타났는지에 대한 정보를 의미한다. n-gram/2L 역색인은 n-gram 역색인에 존재하던 위치 정보의 중복을 제거함으로써 구성된다. 위치 정보의 중복을 제거하기 위한 핵심 아이디어는 n-gram들의 위치 정보를 두 단계로 구성하여 색인하는 것이다. 이를 위해 연속적인 n-gram들을 하나의 subsequence로 묶고, (1) subsequence의 문서 상에서의 위치 정보와 (2) n-gram의 subsequence상에서의 위치 정보로서 n-gram들의 위치 정보를 표현한다.

n-gram/2L 역색인의 주요 장점은 다음과 같다. 첫째, 색인의 크기 측면에서 n-gram/2L 역색인은 색인하려는 데이타의 크기가 클 수록 n-gram 역색인에 비해 유리하다. 즉, 기존의 n-gram 역색인의 크기에 비한 n-gram/2L 역색인의 상대적 크기는 데이타의 크기가 커질수록 작아진다. 둘째, 질의 처리 성능 측면에서도 n-gram/2L 역색인은 색인하려는 데이타의 크기가 클 수록 n-gram 역색인에 비해 유리하다. 즉, 기존의 n-gram 역색인에 비한 n-gram/2L 역색인의 상대적인 질의 처리 성능은 데이타의 크기가 커질수록 좋아진다. 셋

째, n-gram/2L 역색인은 질의 문자열의 길이가 길어져도 질의 처리 시간이 거의 증가하지 않는다. 이들 n-gram/2L 역색인의 공간 복잡도, 시간 복잡도 및 질의 문자열 길이의 영향에 대한 분석은 제5장에서 제시한다.

본 논문의 구성은 다음과 같다. 제2절에서는 기존의 n-gram 역색인의 구조와 그 응용들에 대해 살펴본다. 제3절에서는 본 연구를 수행하게 된 연구 동기를 설명한다. 제4절에서는 n-gram/2L 역색인의 구조와 그 색인 구조에서의 질의 처리 방법을 제안한다. 제5절에서는 n-gram/2L 역색인의 이론적 모델을 설명하고, 그 크기와 성능을 정형적으로 분석한다. 제6절에서는 n-gram/2L 역색인의 성능 평가 결과를 제시한다. 성능 평가를 위하여 Odysseus ORDBMS[10]의 역색인을 사용한다. 마지막으로, 제7절에서는 결론을 내린다.

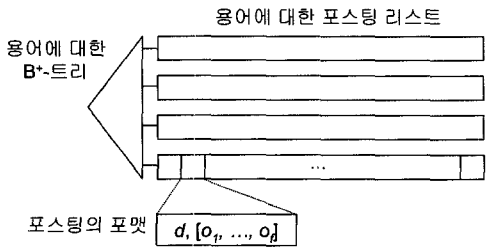
2. 관련 연구

본 장에서는 역색인[1]에 대해 설명하고, 역색인의 한 종류인 n-gram 역색인에 대해 설명한다. **역색인**은 질의로 주어진 용어를 포함하고 있는 문서들을 빠르게 검색하는 색인이다. 여기서 **문서**는 문자들의 한정된 sequence이고, **용어**는 문서의 subsequence이다.

역색인은 기본적으로 용어들에 대한 포스팅 리스트들로 구성된다[4]. 하나의 **포스팅 리스트**는 하나의 용어와 연관되며, 그 용어를 포함하는 문서의 식별자와 용어가 문서에서 나타난 위치 정보들이 리스트 구조로 관리된다. 여기서, 리스트를 구성하는 각 엘리먼트인 문서 식별자와 위치 정보들을 묶어서 **포스팅**이라고 부른다. 용어 t에 대한 포스팅 리스트를 구성하는 과정은 각 문서들에 대해 용어 t가 문서 식별자가 d인 문서에서 오프셋 $o_1,...,o_f$에 f번 발생할 때, 용어 t의 포스팅 리스트에 포스팅 $<d, [o_1,...,o_f]>$을 추가함으로써 수행된다[8]. 일반적으로 질의 처리를 용이하게 하기 위해 포스팅 리스트 내에서 포스팅들은 문서 식별자 d가 증가하는 순서대로 정렬되며, 포스팅 내에서 오프셋들은 오프셋 o가 증가하는 순서대로 정렬된다. 또한, 포스팅 리스트를 빠르게 검색하기 위해 B+-트리 등의 색인이 용어들에 대해 구성된다. 그림 1은 이러한 역색인의 구조를 보여준다.

역색인은 용어의 종류에 따라 단어를 용어로 사용한 단어 기반의 역색인과 n-gram을 용어로 사용한 n-gram 기반의 역색인으로 구분된다[4,6,9]. 이들 중 n-gram 역색인이 본 논문에서 다루고자 하는 역색인이다.

n-gram 역색인은 n-gram을 용어로서 추출하여 구성된 역색인이다. n-gram은 문서 d가 문자의 sequence $c_0,c_1,...,c_{N-1}$로 주어졌을 때, d로부터 추출된 n개의 연속



d: 문서 식별자
o_i: 문서 d에 용어 t가 나타난 오프셋
f: 문서 d에 용어 t가 나타난 횟수

그림 1 역색인의 구조

적인 문자들로 이루어진 문자열이다[11]. n-gram 역색인을 구성하기 위해 주어진 문서 d로부터 n-gram을 빠짐없이 추출하는 방법은 n개의 연속적인 문자들로 이루어진 윈도우를 1-슬라이딩 방식으로 c_0부터 c_{N-n}까지 슬라이딩시키면서 윈도우 내에 위치한 문자열을 추출하는 것이다. 따라서 d의 i번째 n-gram은 문자열 $c_i,c_{i+1},...,c_{i+n-1}$이 된다.

예제 1. 그림 2는 "string"과 "data"라는 문자열이 포함된 문서 집합에 대해 구성된 3-gram 역색인의 포스팅 리스트들의 예이다. "string"에서는 "str", "tri", "rin", "ing"의 네 개의 3-gram이 추출되며, "data"에서는 "dat", "ata"의 두 개의 3-gram이 추출된다. 같은 문자열에서 추출된 3-gram들이 각 문서에서 발생한 위치는 n-gram을 추출하는 방법의 특징으로 인해 오프셋이 1씩 차이남을 알 수 있다. □

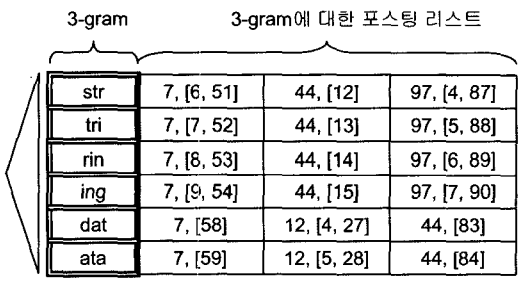


그림 2 3-gram 역색인의 포스팅 리스트들의 예

n-gram 역색인을 사용하여 질의를 처리하기 위해서는 우선 질의 문자열을 n-gram들로 분할한 후 역색인에서 그 n-gram들의 포스팅 리스트들을 찾아 병합(merge)한다[1]. 예를 들어, 그림 2의 n-gram 역색인에서 문자열 "string"을 포함하고 있는 문서들을 찾는 질의는 다음 두 단계를 따른다. 첫 번째 단계에서는 질의 문자열을 "str", "tri", "rin", "ing"의 네 개의 3-gram들로 분할한 후 역색인에서 각 3-gram들을 찾는다. 두

번째 단계에서는 각 3-gram에 대한 포스팅 리스트를 문서 식별자로 합병 조인(merge join)하면서, "str", "tri", "rin", "ing"와 같이 네 개의 3-gram들이 연속해서 나타나 "string"을 구성하는 문서를 구한다. 스캔한 포스팅 리스트 상의 모든 문서에서 네 개의 3-gram들이 연속해서 나타나므로 질의 결과는 문서 식별자 7, 44, 97이 된다.

3. 연구 동기

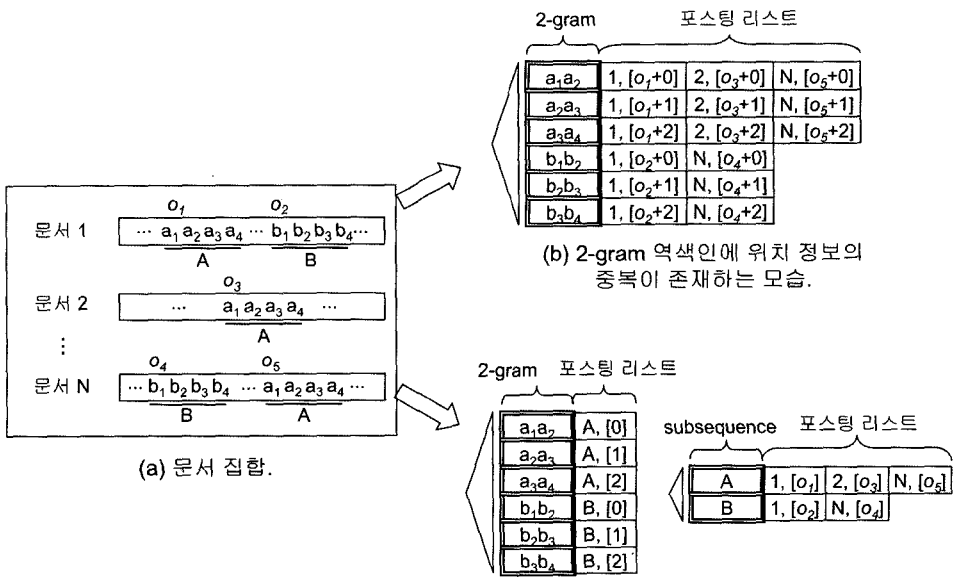
본 장에서는 n-gram 역색인의 크기가 커지는 원인을 분석하고, 그 주요 원인이 n-gram의 위치 정보의 중복 때문임을 밝힌다. 그리고, 그 중복을 제거하는 방안에 대해 설명한다. 그림 3은 설명에 사용할 예제로서 주어진 문서 집합에 대해 2-gram 역색인을 구성한 모습과 2-gram 역색인을 2단계로 구성한 모습을 나타낸다.

그림 3(a)는 N개의 문서들로 구성되어 있는 문서 집합의 예이다. 문자열 "$a_1a_2a_3a_4$"를 subsequence A라 할 때, A는 문서 1, 2, N의 오프셋 o_1, o_3, o_5에 나타나고, 문자열 "$b_1b_2b_3b_4$"를 subsequence B라 할 때, B는 문서 1, N의 오프셋 o_2, o_4에, 나타난다. 따라서, 세 개의 연속적인 2-gram "a_1a_2", "a_2a_3", "a_3a_4"가 문서 1, 2, N에 나타나고, 세 개의 연속적인 2-gram "b_1b_2", "b_2b_3", "b_3b_4"가 문서 1, N에 나타난다.

그림 3(b)는 그림 3(a)로부터 구성된 2-gram 역색인을 나타내며, 회색으로 칠해진 부분은 위치 정보의 중복을 나타낸다. 즉, 그림 3(b)의 2-gram 역색인에서 2-gram "a_1a_2", "a_2a_3", "a_3a_4"의 포스팅 리스트에는 2-gram "a_2a_3", "a_3a_4"의 오프셋이 2-gram "a_1a_2"의 오프셋보다 각각 1, 2만큼씩 크다는 정보가 문서 1 외에 문서 2, N에도 반복해서 나타난다. 2-gram "b_1b_2", "b_2b_3", "b_3b_4"의 포스팅 리스트에도 마찬가지이다. 즉, 문서에서 자주 발생하는 subsequence가 있을 경우 그 subsequence로부터 추출되는 n-gram들의 subsequence 내에서의 상대적인 위치 정보들이 여러 번 반복되어 색인된다.

만약 자주 발생하는 subsequence로부터 추출되는 n-gram들의 상대적인 위치 정보들을 단 한 번만 나타낼 수 있다면, n-gram 역색인에 존재하는 이러한 중복을 제거하여 색인의 크기를 줄일 수 있다. 그림 3(c)는 2-gram 역색인을 2단계로 구성한 모습을 나타낸다. 세 개의 2-gram "a_1a_2", "a_2a_3", "a_3a_4"의 문서 상에서의 위치 정보는 그림 3(c)의 오른쪽과 같이 subsequence A의 문서 상에서의 위치 정보와 그림 3(c)의 왼쪽과 같이 2-gram "a_1a_2", "a_2a_3", "a_3a_4"의 subsequence A 상에서의 위치 정보로 두 단계에 걸쳐 표현함으로써 중복된 정보를 제거할 수 있다.

이와같이 n-gram 역색인을 2단계로 구성하는 것은 릴레이션에 의미 있는(non-trivial) 다치 종속성(MVD: Multivalued Dependency)이 존재할 때 정규화를 통해 두 개의 릴레이션으로 분해 하여 중복을 제거하는 것과



(a) 문서 집합.

(b) 2-gram 역색인에 위치 정보의 중복이 존재하는 모습.

(c) 2-gram 역색인을 2단계 구성함으로써 위치 정보의 중복을 제거한 모습.

그림 3 n-gram 역색인에 존재하는 위치 정보의 중복과 중복의 제거 방법

동일하다. 이를 제5.1절의 정리 2에서 정형적으로 증명한다.

4. n-gram/2L 역색인

4.1 색인 구조

n-gram/2L 역색인은 n-gram들의 위치 정보를 두 단계로 구성한 색인 구조이다. 그림 4는 n-gram/2L 역색인의 구조를 보인다. 이는 백-엔드 역색인과 프런트-엔드 역색인의 계층적 관계의 두 역색인들로 구성된다. **백-엔드(back-end) 역색인** 은 각 subsequence들에 대해 문서 상에서 해당 subsequence가 나타난 절대적인 위치 정보들을 저장하고, **프런트-엔드(front-end) 역색인**은 각 n-gram들에 대해 subsequence 상에서 해당 n-gram이 나타난 상대적인 위치 정보들을 저장한다.

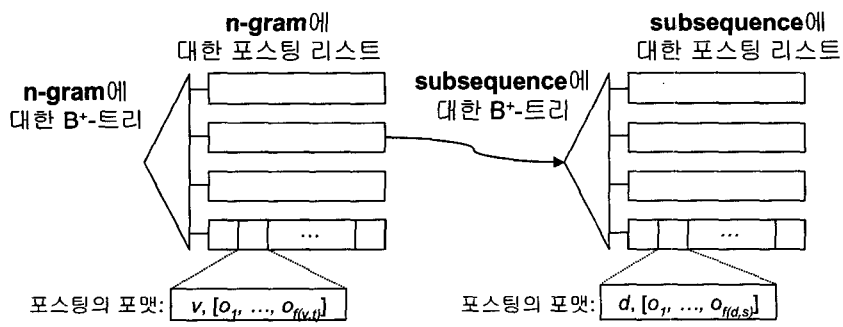
4.2 색인 구성

주어진 문서들의 집합으로부터 n-gram/2L 역색인을 구성하기 위해서는 (1) subsequence 추출 과정, (2) 백-엔드 역색인 구성 과정, (3) n-gram 추출 과정, (4) 프런트-엔드 역색인 구성 과정을 거친다. Subsequence 추출 과정에서는 subsequence의 길이를 m으로 고정하고, subsequence들이 서로 n-1만큼씩 겹치도록 추출한다. Subsequence들을 서로 n-1만큼씩 겹치도록 하는 이유는 n-gram 추출 과정에서 n-gram이 누락되거나 중복되는 것을 방지하기 위해서이며, 이 방법의 정확성은 정리 1에서 증명한다. 이후부터는 서로 n-1만큼씩 겹치도록 추출되는 길이 m의 subsequence를 *m-subsequence*라 부른다. 또한, n은 n-gram의 길이를, m은 m-subsequence의 길이를 나타낸다.

정리 1. Subsequence 추출 과정에서 subsequence들을 서로 n-1만큼씩 겹치도록 하면 n-gram 추출 과정에서 누락되거나 중복되는 n-gram이 발생하지 않는다.

증명. 부록 A 참조. □

그림 5는 n-gram/2L 역색인의 색인 구성 알고리즘 *n-Gram/2L Index Building*을 나타낸다. 단계 1에서는 주어진 문서들의 집합으로부터 m-subsequence들의 집합을 추출한다. 문서들로부터 m-subsequence들을 추출하는 방법은 각 문서를 길이 m의 subsequence들로 나누되, 정리 1에 따라 인접한 두 subsequence가 서로 n-1만큼씩 겹치게 나눈다. 즉, 각 문서가 문자의 sequence $c_0,c_1,...,c_{N-1}$로 이루어질 때, $0 \le i < \lfloor (N-n+1)/(m-n+1) \rfloor$인 모든 i에 대해 문자 $c_{i*(m-n+1)}$을 시작으로 하는 길이 m의 subsequence들을 추출한다. 만약, 마지막 m-subsequence의 길이가 m보다 작다면 문자열의 뒷부분에 공백 문자들을 덧붙여서 길이가 m이 되도록 만든다. 단계 2에서는 단계 1에서 얻은 m-subsequence 집합으로부터 문서에 대한 백-엔드 역색인을 구성한다. 즉, 각 m-subsequence s에 대해 s가 식별자 d인 문서에서 f번 발생하였고, 그 위치가 문서의 오프셋 $o_1,...,o_f$이라면, s와 연관된 포스팅 리스트에 포스팅 $<d, [o_1,...,o_f]>$을 추가한다. 단계 3에서는 단계 1에서 얻은 m-subsequence 집합으로부터 n-gram 집합을 추출한다. 각 m-subsequence에 대해 1-슬라이딩 방식으로 n-gram들을 추출한다. 단계 4에서는 단계 3에서 얻은 n-gram 집합으로부터 m-subsequence에 대한 프런트-엔드 역색인을 구성한다. 즉, 각 n-gram g에 대해 g가 식별자 v인 m-subsequence에서 f번 발생하였고, 그 위



(a) 프런트-엔드 역색인. (b) 백-엔드 역색인.

그림 4 n-gram/2L 역색인의 구조

***n-Gram/2L Index Building* 알고리즘:**

입력: (1) 문서 집합 D, (2) m-subsequence의 길이 m, (3) n-gram의 길이 n

출력: n-gram/2L 역색인

알고리즘:

과정 1. m-subsequence 추출 단계: 주어진 문서 집합 D의 각 문서에 대해 다음을 수행한다.

1.1 각 문서가 문자의 시퀀스 $c_0,c_1,...,c_{N-1}$로 이루어질 때,

$c_{i*(m-n+1)}$ $(0 \le i < \lfloor (N-n+1)/(m-n+1) \rfloor)$을 시작으로 하는 길이 m의 m-subsequence들을 추출하고,

그 m-subsequence의 문서에서의 오프셋들을 기록한다.

1.2 마지막 m-subsequence의 길이가 m보다 작다면 m-subsequence의 뒷부분에

공백 문자들을 덧붙여서 길이가 m이 되도록 만든다.

과정 2. 백-엔드 역색인 구성 단계: 과정 1에서 구한 각 m-subsequence에 대해 다음을 수행한다.

2.1 m-subsequence s가 문서 d의 오프셋 $o_0,o_1,...,o_f$에서 발생했다면,

s의 포스팅 리스트에 $<d, [o_0,o_1,...,o_f]>$를 추가한다.

과정 3. n-gram 추출 단계: 과정 1에서 구한 각 m-subsequence에 대해 식별자를 부여하고 다음을 수행한다.

3.1 m-subsequence s가 문자의 시퀀스 $c_0,c_1,...,c_{L-1}$로 이루어질 때,

c_i $(0 \le i < L-n+1)$을 시작으로 하는 길이 n의 n-gram들을 추출하고,

그 n-gram의 s 내에서의 오프셋들을 기록한다.

과정 4. 프런트-엔드 역색인 구성 단계: 단계 3에서 구한 n-gram 집합의 각 n-gram에 대해 다음을 수행한다.

4.1 n-gram g가 m-subsequence v의 오프셋 $o_0,o_1,...,o_f$에서 발생했다면,

g의 포스팅 리스트에 $<v, [o_0,o_1,...,o_f]>$를 추가한다.

그림 5 n-gram/2L 역색인의 색인 구성 알고리즘

치가 m-subsequence의 오프셋 $o_1,...,o_f$이라면, g와 연관된 포스팅 리스트에 포스팅 $<v, [o_1,...,o_f]>$을 추가한다.

예제 2. 그림 6은 주어진 문서들의 집합으로부터 n-gram/2L 역색인을 구성한 예이다. 여기에서 n = 2이며, m = 4이다. 그림 6(a)는 주어진 문서 집합을 나타낸다. 그림 6(b)는 단계 1에서 그림 6(a)의 문서 집합으로부터 추출된 4-subsequence들의 집합을 나타낸다. 문서들은 길이가 4이고 서로 1 (즉, n-1)만큼씩 겹치는 4-subsequence들로 나눠지므로 문서 0으로부터 추출되는 4-subsequence들은 "ABCD", "DDAB", "BBCD"이다. 그림 6(c)는 단계 2에서 추출된 4-subsequence 집합으로부터 구성된 백-엔드 역색인이다. 4-subsequence "ABCD"는 문서 0, 3, 4에서 각각 0, 3, 6인 오프셋에서 나타났으므로 4-subsequence "ABCD"와 연관된 포스팅 리스트에는 포스팅 <0, [0]>, <3, [3]>, <4, [6]>이 추가된다. 그림 6(d)는 그림 6(b)의 4-subsequence에 부여된 식별자이고, 그림 6(e)는 단계 3에서 4-subsequence 집합으로부터 추출된 2-gram 집합을 나타낸다. 1-슬라이딩 방식으로 2-gram들을 추출하므로 4-subsequence 0으로부터 추출되는 2-gram들은 "AB", "BC", "CD"이다. 그림 6(f)는 단계 4에서 2-gram 집합으로부터 구성된 프런트-엔드 역색인이다. 2-gram "AB"는 4-subsequence 0, 3, 4, 5에서 각각 0, 2, 1, 2인 오프셋에서 나타났으므로 2-gram "AB"와 연관된 포스팅 리스트에는 포스팅 <0, [0]>, <3, [2]>, <4, [1]>, <5, [2]>가 추가된다. □

4.3 질의 처리

본 절에서는 n-gram/2L 역색인을 사용한 질의 처리 알고리즘을 설명한다. 본 절에서는 설명의 편의를 위해 하나의 문자열로 된 정확 문자열 매칭(exact string matching) 질의를 다룬다. 본 방법은 근사 문자열 매칭(approximate string matching) 질의나 여러 개의 문자열로 구성된 불(Boolean) 질의를 처리하는 방법으로도 확장 가능하다.

n-gram/2L 역색인을 사용한 질의 처리는 두 단계로 구성된다. 즉, 먼저 프런트-엔드 역색인을 검색하여 얻어진 결과에 한해서, 다시 백-엔드 역색인을 검색하여 최종 결과를 얻는다. 첫 번째 단계에서는 질의 문자열을 분할하여 얻은 n-gram들을 가지고 프런트-엔드 역색인을 검색함으로써 주어진 질의 문자열을 **커버**하는

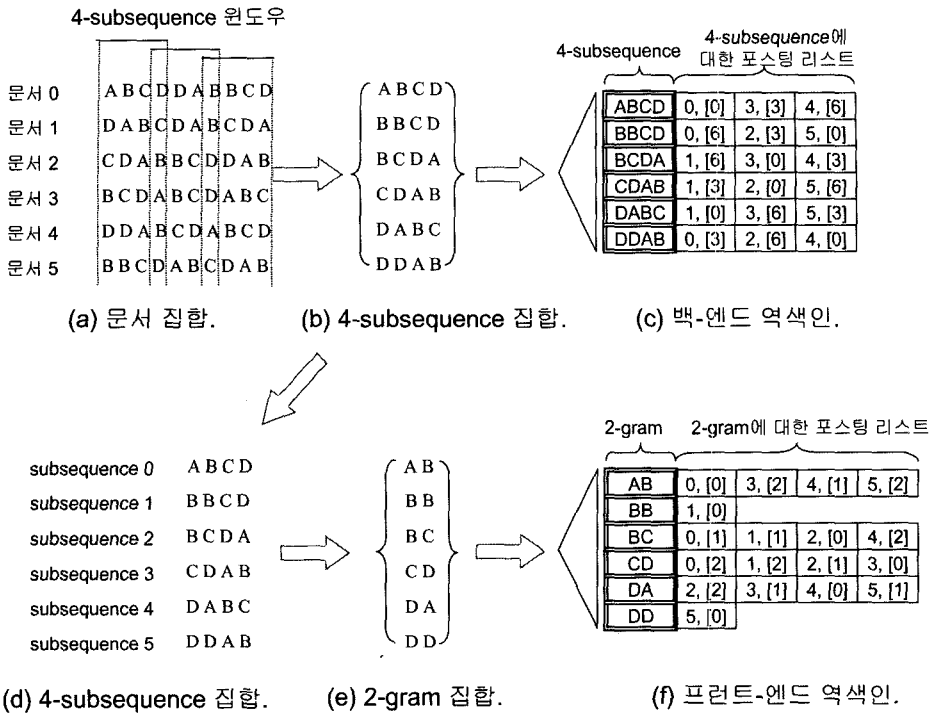


그림 6 n-gram/2L 역색인 구성의 예

m-subsequence들을 찾아낸다. 이 단계에서 질의 문자열을 커버하지 않는 m-subsequence들은 필터링 된다. 두 번째 단계에서는 첫 번째 단계에서 얻은 m-subsequence들을 가지고 백-엔드 역색인을 검색함으로써 질의 문자열을 **포함**하는 m-subsequence들의 집합 $\{S_i\}$를 가지는 문서들을 찾아낸다. 첫 번째 단계에서 구한 m-subsequence들을 하나 이상 포함하는 문서들은 질의의 필요 조건을 만족하는 후보 문서 집합이며, 두 번째 단계에서는 그 문서 집합으로부터 착오 해답인 문서들을 제거함으로써 정제작업(refinement)을 수행한다.

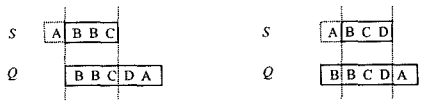
이제는 정의 1에서 **커버(cover)**를 정형적으로 정의하고, 정의 3에서 **포함(contain)**을 정형적으로 정의한다.

정의 1. m-subsequence S가 질의 문자열 Q에 대해 다음의 네 가지 조건 중 하나를 만족하면 S가 Q를 **커버**한다고 정의한다: (1) S의 서픽스가 Q의 프리픽스와 일치하거나, (2) S 전체가 Q의 서브스트링과 일치하거나, (3) S의 프리픽스가 Q의 서픽스와 일치하거나, (4) S의 서브스트링이 Q 전체와 일치한다. □

정의 2. 문자열들의 sequence를 하나의 문자열로 전개하는 함수 *expand*를 다음과 같이 정의한다: (1) 두 개의 문자열 S_i, S_p가 각각 문자열 $c_i...c_j$와 $c_p...c_q$로 주어지고, S_i의 서픽스와 S_p의 프리픽스가 $c_{j-k+1}...c_j$ = $c_p...c_{p+k-1}$과 같이 k만큼 겹쳐져 있다고 하자 ($k \geq 0$). 이 때, $expand(S_iS_p) = c_i...c_jc_{p+k}...c_q$이다. (2) 두 개 이상의 문자열 S_l, S_{l+1},..., S_m에 대해 $expand(S_lS_{l+1}...S_m) = expand(expand(S_lS_{l+1}),S_{l+2}...S_m)$이다. □

정의 3. m-subsequence들의 집합 $\{S_i\}$가 다음의 조건을 만족하면, $\{S_i\}$가 Q를 **포함**한다고 정의한다: $\{S_i\}$에 속한 m-subsequence들이 겹치도록 연속적으로 연결된 sequence를 $S_lS_{l+1}...S_m$이라 할 때, $expand(S_lS_{l+1}...S_m)$의 서브스트링이 질의 문자열 Q 전체와 일치한다. □

예제 3. 그림 7은 질의 문자열 Q를 커버하는 m-subsequence의 예와 커버하지 않는 m-subsequence의 예를 보여준다. 그림 7(a)에서는 S의 서픽스가 Q의 프리픽스와 일치하므로 S는 Q를 커버한다. 그림 7(b)에서는 S의 서브스트링 "BCD"가 정의 1의 네 가지 조건을 모두 만족하지 않기 때문에 S는 Q를 커버하지 않는다. □



(a) S가 Q를 커버함. (b) S가 Q를 커버하지 않음.

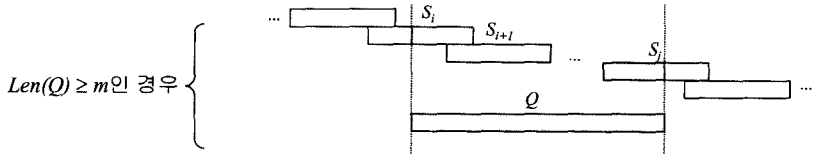
그림 7 m-subsequence S가 질의 문자열 Q를 커버하는 예와 커버하지 않는 예

보조정리 1. 질의 문자열 Q를 포함하는 m-subsequence들의 집합 $\{S_i\}$가 추출된 문서는 Q를 커버하는 m-subsequence를 하나 이상 포함한다.

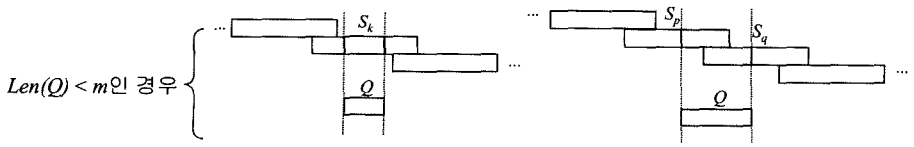
증명. 우선 m-subsequence들의 집합 $\{S_i\}$가 Q를 포함하는 모든 경우들을 나타내면 그림 8과 같다. 이때, Q의 길이 $Len(Q)$가 m보다 큰 경우인 그림 8(a)와 작은 경우인 그림 8(b), 그림 8(c)로 나누어서 고려한다. 그림 8(a)에서는 $(j-i+1)$개의 m-subsequence들의 집합 $\{S_i, ...,S_j\}$가 Q를 포함한다. 그림 8(b)에서는 한 개의 m-subsequence S_k가 Q를 포함한다. 그림 8(c)에서는 두 개의 m-subsequence들의 집합 $\{S_p,S_q\}$이 Q를 포함한다. 이와 같이 m-subsequence들의 집합 $\{S_i\}$가 Q를 포함한다면, $\{S_i\}$에 속한 각 m-subsequence는 Q를 커버함을 알 수 있다. 따라서, 그러한 m-subsequence들의 집합 $\{S_i\}$가 추출된 문서는 Q를 커버하는 m-subsequence를 최소한 하나 이상 포함하게 된다. □

그림 9는 n-gram/2L 역색인을 사용한 질의 처리 알고리즘 *n-Gram/2L Index Searching*을 나타낸다. 단계 1에서는 질의 문자열 Q에서 n-gram들을 추출하여 프런트-엔드 역색인에서 각 n-gram들을 검색한다. 그리고, 각 n-gram에 대한 포스팅 리스트들을 m-subsequence 식별자로 합병 외부 조인(merge outer join)하면서, 정의 1에 따라 Q를 커버하는(즉, 보조정리 1의 필요 조건을 만족하는) m-subsequence들을 집합 S_{cover}에 추가한다. Q를 커버하는 m-subsequence는 일반적으로 Q로부터 추출된 모든 n-gram들을 가지고 있지 않기 때문에 질의 처리 알고리즘의 단계 1.2에서 합병 외부 조인을 수행한다. 이때, m-subsequence가 Q를 커버하는지는 합병 외부 조인되는 포스팅들이 가지고 있는 오프셋 정보를 이용하여 확인한다. 단계 2에서는 집합 S_{cover}에 포함된 m-subsequence들에 대한 포스팅 리스트들을 문서 식별자로 합병 외부 조인하면서, 같은 문서 식별자 d_i를 가지는 m-subsequence들의 집합 $\{S_i\}$가 정의 3에 따라 Q를 포함하는지를 검사하는 정제작업을 수행한다. 이때, 집합 $\{S_i\}$가 Q를 포함하는지는 합병 외부 조인되는 포스팅들이 가지고 있는 오프셋 정보를 이용하여 확인한다. 만약, 집합 $\{S_i\}$가 Q를 포함한다면 d_i를 질의 결과로 반환한다.

예제 4. 예제 3의 n-gram/2L 역색인에 대해 질의 "ABCD"를 처리하는 과정은 다음 두 단계로 수행된다. 첫 번째 단계에서는 질의 문자열 "ABCD"를 2-gram "AB", "BC", "CD"로 분할하여 프런트-엔드 역색인을 검색하면서 정의 1에 따라 질의 문자열 "ABCD"를 cover하는 4-subsequence들을 집합 S_{cover}에 추가한다. 4-subsequence 0은 정의 1의 조건 (2)를, 4-subsequence 2는 정의 1의 조건 (3)을, 4-subsequence 3은 정의 1의 조건 (1)과 (3)을, 4-subsequence 4와 5는 정의 1의 조건 (1)을 만족하므로 S_{cover}에는 4-subsequence 0, 2, 3, 4, 5가 추가된다. 두 번째 단계에서는 S_{cover}에 포함된 4-subsequence "ABCD"(0), "BCDA"(2), "CDAB"(3), "DABC"(4), "DDAB"(5)에 대한 포스팅 리스트들을 문서 식별자로 합병 외부 조인하면서 동일한 문서로부터 추출된 4-subsequence들이 정의 3에 따라 질의 문자열 "ABCD"를 contain하는지를 검사한다. 문서 0으로부터 추출된 4-subsequence "ABCD", "DDAB"는 오프셋 정보를 이용하여 정의 2의 expand 함수를 적용시킬 경우 문자열 "ABCDDAB"가 되고, 이 문자열의 서브스트링은 질의 문자열과 일치하므로, 문서 0으로부터 추출된 4-subsequence들은 정의 3에 따라 질의 문자열을 contain한다. 마찬가지로 문서 1, 3, 4, 5로부터 추출된 4-subsequence들도 질의 문자열을 contain한다. 따라서, 질의 결과로서 문서 식별자 0, 1,



(a) $\{S_i, S_{i+1}, ... S_j\}$가 Q를 포함한다.



(b) $\{S_k\}$가 Q를 포함한다. (c) $\{S_p, S_q\}$가 Q를 포함한다.

그림 8 m-subsequence들의 집합이 Q를 포함하는 경우들

***n-Gram/2L Index Searching* 알고리즘:**
입력: (1) n-gram/2L 역색인
(2) 질의 문자열 Q
출력: Q를 포함하는 문서 식별자들
알고리즘:
과정 1. 프런트-엔드 역색인 검색 단계:
1.1 Q를 n-gram들로 분할하여 프런트-엔드 역색인에서 각 n-gram들을 검색한다.
1.2 각 n-gram들에 대한 포스팅 리스트들을 m-subsequence 식별자로 합병 외부 조인하면서, 정의 1에 따라 Q를 cover하는 m-subsequence들을 집합 S_{cover}에 추가한다.
과정 2. 백-엔드 역색인 검색 단계:
2.1 S_{cover}에 포함된 m-subsequence들에 대한 포스팅 리스트들을 문서 식별자로 합병 외부 조인하면서 다음을 수행한다.
2.1.1 동일한 문서 d_i로부터 추출된 m-subsequence들의 집합 $\{S_i\}$가 정의 3에 따라 Q를 contain하는지를 검사한다.
2.1.2 $\{S_i\}$가 Q를 contain하면 d_i를 질의 결과로 반환한다.

그림 9 n-gram/2L 역색인의 질의 처리 알고리즘

3, 4, 5가 반환된다. □

5. n-gram/2L 역색인의 이론적 분석

본 장에서는 n-gram/2L 역색인에 대한 이론적 분석 결과를 제시한다. 제5.1절에서는 n-gram 역색인의 중복을 제거함으로써 n-gram/2L 역색인이 도출됨을 정형적으로 증명한다. 제5.2절에서는 n-gram 역색인과 n-gram/2L 역색인의 크기를 분석하여 각각의 공간 복잡도를 구하고, 제5.3절에서는 두 역색인의 질의 처리 성능을 분석하여 각각의 시간 복잡도를 구한다.

5.1 n-gram/2L 역색인의 정형화

본 절에서는 먼저 n-gram 역색인에 존재하는 위치 정보의 중복이 의미 있는 다치 종속성[12,13]로 인한 것임을 보이고, 다음으로 제4정규형(4NF: Fourth Normal Form)을 따르도록 릴레이션을 분해하는 방법을 통해 그 중복을 제거함으로써 n-gram/2L 역색인이 도출됨을 보인다.

이론적 전개를 위하여 우선 n-gram 역색인을 제1정규형(1NF: First Normal Form)으로 바꾼 릴레이션을 고려한다. 이러한 릴레이션을 **NDO 릴레이션**이라 부른다. 이 릴레이션은 N, D, O 세 개의 애트리뷰트로 구성된다. 여기에서 N은 n-gram을, D는 문서 식별자를, O는 오프셋을 나타낸다. 그리고 NDO 릴레이션에 애트리뷰트 S를 추가하고, 애트리뷰트 O를 애트리뷰트 O_1, O_2로 나누어 총 다섯 개의 애트리뷰트 S, N, D, O_1, O_2를 가지도록 표현한 릴레이션을 **$SNDO_1O_2$ 릴레이션**이라고 부른다. 여기에서 S는 n-gram이 추출되는 길이 m의 m-subsequence를, O_1은 n-gram이 나타난 m-subsequence 상에서의 오프셋을, O_2는 m-subsequence가 나타난 문서 상에서의 오프셋을 나타낸다.

$SNDO_1O_2$ 릴레이션의 애트리뷰트 S, O_1, O_2의 값은 NDO 릴레이션의 애트리뷰트 N, D, O의 값에 의해 자동으로 결정된다. 주어진 문서 집합의 각 문서가 m-subsequence들로 n-1만큼씩 겹치도록 나누어져 있다고 가정하자. 그러면, 정리 1에 의해 문서 집합으로부터 추출되는 임의의 n-gram은 하나의 m-subsequence에만 속하게 된다. NDO 릴레이션의 임의의 투플 (n, d, o)으로부터 자동으로 결정되는 $SNDO_1O_2$ 릴레이션의 투플 (s, n, d, o_1, o_2)에서, s는 문서 d의 오프셋 o에 있는 n-gram n이 속해 있는 m-subsequence이다. 그리고, o_1은 s상에서 n-gram n이 나타난 오프셋이며, o_2는 문서 d상에서 s가 나타난 오프셋이다.

예제 5. 그림 10은 그림 6(a)의 문서 집합으로부터 2-gram 역색인을 구성한 예이다. 그림 11(a)는 그림 10의 2-gram 역색인을 NDO 릴레이션으로 표현한 예이며, 그림 11(b)는 m = 4일 때 그림 11(a)의 NDO 릴레이션을 $SNDO_1O_2$ 릴레이션으로 표현한 후 애트리뷰트 S의 값에 의해 정렬한 예이다. 그림 11(b)의 $SNDO_1O_2$ 릴레이션에서 검정색 마커로 표시된 투플은 그림 11(a)의 NDO 릴레이션에서 검정색 마커로 표시된 투플로부터 자동으로 결정된다. 그림 6(a)의 문서 집합에서 문서 0의 오프셋 1에 나타나는 2-gram "BC"가 속해 있는

2-gram	2-gram에 대한 포스팅 리스트					
AB	0, [0, 5]	1, [1, 5]	2, [2, 8]	3, [3, 7]	4, [2, 6]	5, [4, 8]
BB	0, [6]	2, [3]	5, [0]			
BC	0, [1, 7]	1, [2, 6]	2, [4]	3, [0, 4, 8]	4, [3, 7]	5, [1, 5]
CD	0, [2, 8]	1, [3, 7]	2, [0, 5]	3, [1, 5]	4, [4, 8]	5, [2, 6]
DA	0, [4]	1, [0, 4, 8]	2, [1, 7]	3, [2, 6]	4, [1, 5]	5, [3, 7]
DD	0, [3]	2, [6]	4, [0]			

그림 10 n-gram 역색인의 예

N	D	O
AB	0	0
AB	0	5
AB	1	1
AB	1	5
AB	2	2
AB	2	8
AB	3	3
AB	3	7
AB	4	2
AB	4	6
AB	5	4
AB	5	8
BB	0	6
BB	2	3
BB	5	0
BC	0	1
BC	0	7
BC	1	2
BC	1	6
BC	2	4
BC	3	0
BC	3	4
BC	3	8
BC	4	3
BC	4	7
BC	5	1
BC	5	5

CD	0	2
CD	0	8
CD	1	3
CD	1	7
CD	2	0
CD	2	5
CD	3	1
CD	3	5
CD	4	4
CD	4	8
CD	5	2
CD	5	6
DA	0	4
DA	1	0
DA	1	4
DA	1	8
DA	2	1
DA	2	7
DA	3	2
DA	3	6
DA	4	1
DA	4	5
DA	5	3
DA	5	7
DD	0	3
DD	2	6
DD	4	0

(a) NDO 릴레이션의 예.

S	N	O_1	D	O_2
ABCD	AB	0	0	0
ABCD	AB	0	3	3
ABCD	AB	0	4	6
ABCD	BC	1	0	0
ABCD	BC	1	3	3
ABCD	BC	1	4	6
ABCD	CD	2	0	0
ABCD	CD	2	3	3
ABCD	CD	2	4	6
BBCD	BB	0	0	6
BBCD	BB	0	2	3
BBCD	BB	0	5	0
BBCD	BC	1	0	6
BBCD	BC	1	2	3
BBCD	BC	1	5	0
BBCD	CD	2	0	6
BBCD	CD	2	2	3
BBCD	CD	2	5	0
BCDA	BC	0	1	6
BCDA	BC	0	3	0
BCDA	BC	0	4	3
BCDA	CD	1	1	6
BCDA	CD	1	3	0
BCDA	CD	1	4	3
BCDA	DA	2	1	6
BCDA	DA	2	3	0
BCDA	DA	2	4	3

CDAB	CD	0	1	3
CDAB	CD	0	2	0
CDAB	CD	0	5	6
CDAB	DA	1	1	3
CDAB	DA	1	2	0
CDAB	DA	1	5	6
CDAB	AB	2	1	3
CDAB	AB	2	2	0
CDAB	AB	2	5	6
DABC	DA	0	1	0
DABC	DA	0	3	6
DABC	DA	0	5	3
DABC	AB	1	1	0
DABC	AB	1	3	6
DABC	AB	1	5	3
DABC	BC	2	1	0
DABC	BC	2	3	6
DABC	BC	2	5	3
DDAB	DD	0	0	3
DDAB	DD	0	2	6
DDAB	DD	0	4	0
DDAB	DA	1	0	3
DDAB	DA	1	2	6
DDAB	DA	1	4	0
DDAB	AB	2	0	3
DDAB	AB	2	2	6
DDAB	AB	2	4	0

(b) $SNDO_1O_2$ 릴레이션의 예.
(애트리뷰트 S에 의해 정렬됨)

그림 11 $SNDO_1O_2$ 릴레이션에 의미 있는 다치 종속성이 존재함을 보이는 예

4-subsequence는 "ABCD"이므로, 애트리뷰트 S의 값은 "ABCD"로 결정된다. 애트리뷰트 O_1, O_2의 값은 각각 4-subsequence "ABCD"에서 2-gram "BC"가 나타난 오프셋 1과 문서 0에서 4-subsequence "ABCD"가 나타난 오프셋 0으로 결정된다. □

이제 보조정리 2에서 $SNDO_1O_2$ 릴레이션, 즉 n-gram 역색인에 의미 있는 다치 종속성이 존재함을 보인다.

보조정리 2. $SNDO_1O_2$ 릴레이션에는 $S \rightarrow\rightarrow NO_1$와 $S \rightarrow\rightarrow DO_2$의 의미 있는 다치 종속성이 존재한다. 이때, S는 수퍼키가 아니다.

증명. 다치 종속성의 정의[12-14]에 따르면 스키마 R을 따르는 임의의 릴레이션 r에서 μ, $\nu \in r$, $\mu[X] = \nu[X]$인 두 개의 투플 μ, ν가 존재할 때 아래 세 가지 조건을 만족하는 두 개의 투플 ϕ, ψ도 반드시 존재하면, $X \rightarrow\rightarrow Y$의 다치 종속성이 성립한다(X, Y는 R의 부분집합). 그리고, $Y \not\subset X$이고, $X \cup Y \neq R$ (즉, $R-X-Y \neq \varnothing$)(Y와 R-X-Y가 애트리뷰트 공집합이 아니다)이면, 다치 종속성 $X \rightarrow\rightarrow Y$는 의미 있는 다치 종속성이다[12,13]. 즉, 애트리뷰트 X의 임의의 값에 대해 애트리뷰트 Y의 값들과 애트리뷰트 R-X-Y의 값들이 카테시안 곱(Cartesian product) 형태를 이루면 의미 있는 다치 종속성이 성립한다[15].

1. $\phi[X] = \psi[X] = \mu[X] = \nu[X]$
2. $\phi[Y] = \mu[Y]$ and $\phi[R\text{-}X\text{-}Y] = \nu[R\text{-}X\text{-}Y]$
3. $\psi[Y] = \nu[Y]$ and $\psi[R\text{-}X\text{-}Y] = \mu[R\text{-}X\text{-}Y]$

주어진 문서 집합으로부터 추출되는 m-subsequence들의 집합을 $\{S_1,...,S_r\}$라 할때, m-subsequence S_i로부터 추출되는 n-gram들의 집합 $\{N_{i1},...,N_{iq}\}$은 S_i가 나타나는 문서들의 집합 $\{D_{i1},...,D_{ip}\}$로부터도 항상 추출된다 ($1 \le i \le r$). 따라서, $SNDO_1O_2$ 릴레이션에서 애트리뷰트 S의 값이 m-subsequence S_i인 투플들에 대해, S_i로부터 추출된 n-gram들을 나타내는 애트리뷰트 NO_1의 값들과 S_i가 출현한 문서들을 나타내는 애트리뷰트 DO_2의 값들은 항상 카테시안 곱 형태를 이룬다. R = $SNDO_1O_2$, X = S, Y = NO_1, R-X-Y = DO_2이라고 하면, 애트리뷰트 X의 임의의 값에 대해 애트리뷰트 Y의 값들과 애트리뷰트 R-X-Y의 값들이 카테시안 곱 형태를 이루므로 위의 세 가지 조건이 만족된다. Y = DO_2라고 하여도 마찬가지의 사실이 성립한다. 또한, $NO_1 \not\subset S$, $DO_2 \not\subset S$, $NO_1 \cup S \neq SNDO_1O_2$, $DO_2 \cup S \neq SNDO_1O_2$이다. 따라서, $SNDO_1O_2$ 릴레이션에는 $S \rightarrow\!\!\!\!\rightarrow NO_1$과 $S \rightarrow\!\!\!\!\rightarrow DO_2$의 의미 있는 다치 종속성이 존재한다. 이때, 그림 11(b)의 반대 예제에서 보여지듯이 S는 수퍼키가 아니다. □

$SNDO_1O_2$ 릴레이션에 의미 있는 다치 종속성이 존재하는 이유를 직관적으로 설명하면 임의의 *m-subsequence*가 어떤 문서들에 나타나는가와 그 m-subsequence로부터 어떤 n-gram들이 추출되는가는 서로 무관계이기 때문이다. 이렇게 서로 무관계인 애트리뷰트들이 하나의 릴레이션에 같이 있을 경우에는 다치 종속성이 존재하게 된다[12-15]. $SNDO_1O_2$ 릴레이션에서는 문서와 n-gram 사이의 무관계성으로 인해 문서 식별자와 n-gram의 가능한 모든 조합을 표현하는 투플들이 각 *m-subsequence*마다 유지된다.

예제 5. 그림 11(b)는 $SNDO_1O_2$ 릴레이션에 의미 있는 다치 종속성 $S \rightarrow\!\!\!\!\rightarrow NO_1$와 $S \rightarrow\!\!\!\!\rightarrow DO_2$가 존재하는 예이다. 그림 11(b)의 $SNDO_1O_2$ 릴레이션에서 애트리뷰트 S의 값이 "ABCD"인 회색으로 칠해진 부분을 보면, 두 애트리뷰트 N, O_1의 값 ("AB", 0), ("BC", 1), ("CD", 2)에 대해 두 애트리뷰트 D, O_2의 값 (0, 0), (3, 3), (4, 6)이 반복해서 나타나는 중복이 있다. 즉, S="ABCD"일 때 NO_1과 DO_2가 카테시안 곱을 이룬다. 애트리뷰트 S의 나머지 값들에 대해서도 이러한 반복이 나타남을 볼 수 있다. □

따름정리 1. $SNDO_1O_2$ 릴레이션은 제4정규형이 아니다.

증명. 의미 있는 다치 종속성 $S \rightarrow\!\!\!\!\rightarrow NO_1$가 존재하며, S가 수퍼키가 아니다. □

n-gram/2L 역색인의 프런트-엔드 역색인과 백-엔드 역색인은 바로 $SNDO_1O_2$ 릴레이션을 제4정규형을 따르도록 분해함으로써 얻게 되는 두 개의 릴레이션에 해당한다. 이를 정리 2에서 증명한다. 이로인해 n-gram/2L 역색인에는 의미 있는 다치 종속성에 의한 중복이 제거됨이 이론적으로 증명된다[14].

보조정리 3. $SNDO_1O_2$ 릴레이션을 SNO_1과 SDO_2 의 두 개의 릴레이션으로 분해 했을 때, 각각의 릴레이션은 제4정규형이다.

증명. 부록 B 참조. □

정리 2. $SNDO_1O_2$ 릴레이션을 제4정규형을 따르도록 분해한 SNO_1 릴레이션과 SDO_2 릴레이션은 각각 n-gram/2L 역색인의 프런트-엔드 역색인과 백-엔드 역색인이 된다.

증명. SNO_1 릴레이션은 애트리뷰트 N, S, O_1이 각각 용어, m-subsequence 식별자, 위치정보인 프런트-엔드 역색인 형태로 표현될 수 있고, SDO_2 릴레이션은 애트리뷰트 S, D, O_2가 각각 용어, 문서 식별자, 위치정보인 백-엔드 역색인 형태로 표현될 수 있다. 따라서, $SNDO_1O_2$ 릴레이션을 SNO_1 릴레이션과 SDO_2 릴레이션으로 분해한 결과는 n-gram/2L 역색인의 프런트-엔드 역색인과 백-엔드 역색인이 된다. □

예제 6. 그림 12는 그림 11의 $SNDO_1O_2$ 릴레이션이 두 개의 릴레이션 SNO_1과 SDO_2로 분해된 결과이다. SDO_2 릴레이션의 애트리뷰트 S의 값에서 괄호안의 값은 m-subsequence에 부여된 식별자를 나타낸다. 그림 11의 $SNDO_1O_2$ 릴레이션에서 회색으로 칠해진 투플들은 그림 12의 SNO_1 릴레이션과 SDO_2 릴레이션에서의 회색으로 칠해진 투플들로 분해된다. 그림 11에서 보이는 NO_1과 DO_2 사이의 카테시안 곱이 그림 12에서는 제거됨을 알 수 있다. SNO_1 릴레이션을 역색인 형태로 표현하면 그림 6의 프런트-엔드 역색인과 동일해짐을 알 수 있고, SDO_2 릴레이션을 역색인 형태로 표현하면 그림 6의 백-엔드 역색인과 동일해짐을 알 수 있다.

5.2 색인 크기의 분석

n-gram/2L 역색인의 크기에 영향을 미치는 파라미터는 *n-gram*의 길이인 n과 m-subsequence의 길이인 m이다. 이 중에서 n의 값은 일반적으로 응용에 따라 결정되는 수치인 반면, m의 값은 본 논문에서 n-gram/2L 역색인의 구성을 위해 도입한 수치로서 자유롭게 조절할 수 있다. 따라서, 본 절에서는 n-gram/2L 역색인의 크기를 분석하고, 그 크기를 최소로 만드는 최적 길이 m의 값을 결정하기 위한 모델을 제시한다. 이후부터는 최적 길이 m을 m_o라 부른다.

먼저 표 1에서 m_o를 구하기 위해 사용되는 표기법을 정의한다. 여기에서, n-gram 역색인과 n-gram/2L 역

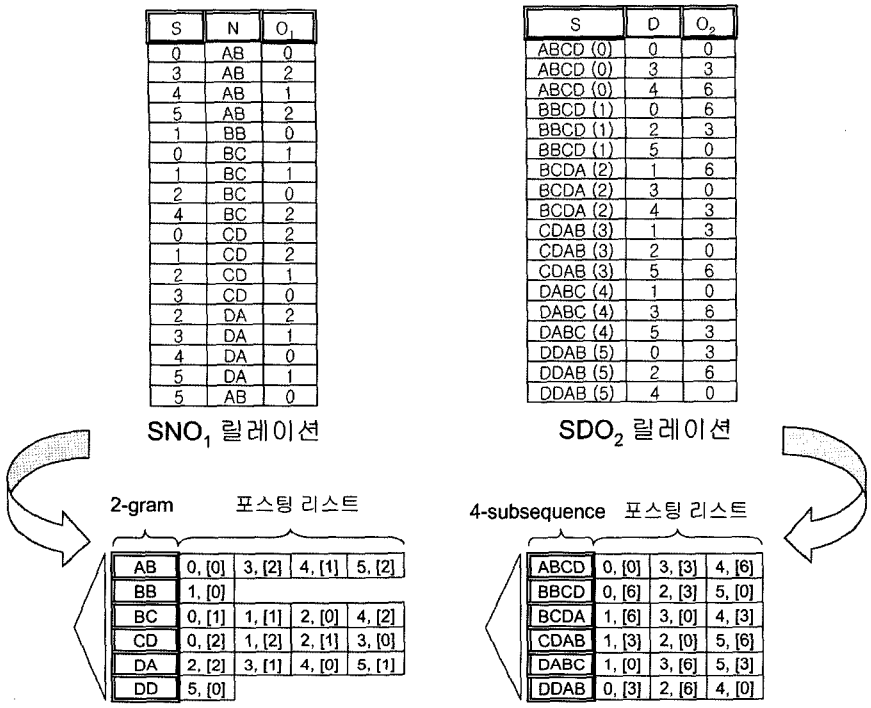


그림 12 그림 11의 $SNDO_1O_2$ 릴레이션이 두 개의 릴레이션으로 분해된 결과

표 1 최적 길이 m_o를 구하기 위해 사용되는 표기법

기 호	정의/의미
$size_{ngram}$	n-gram 역색인의 크기
$size_{front}$	n-gram/2L 역색인에서의 프런트-엔드 역색인의 크기
$size_{back}$	n-gram/2L 역색인에서의 백-엔드 역색인의 크기
S	문서 집합으로부터 추출되는 유일(unique)한 m-subsequence들의 집합
$k_{ngram}(s)$	m-subsequence s에서 추출된 n-gram들의 개수
$k_{doc}(s)$	m-subsequence s가 문서 집합에서 나타난 횟수
$avg_{ngram}(S)$	$s \in S$일 때, $k_{ngram}(s)$의 평균값 (= $(\sum_{s \in S} k_{ngram}(s)) / \mid S \mid$)
$avg_{doc}(S)$	$s \in S$일 때, $k_{doc}(s)$의 평균값 (= $(\sum_{s \in S} k_{doc}(s)) / \mid S \mid$)

색인의 크기로는 색인에 저장된 오프셋들의 개수를 사용한다. 이는 역색인의 크기가 문서상에서 용어가 나타난 오프셋들의 개수에 대체적으로 비례하기 때문이다 [4].

본 절에서는 분할(decomposition)을 통해 색인의 크기가 감소하는 비율이 최대가 되는 m_o의 값을 결정하고자 하며, 이를 위해 정의 4에서 분할 효율을 정의한다.

정의 4. n-gram 역색인의 크기에 대한 n-gram/2L 역색인의 크기의 비율의 역수를 분할 효율이라고 정의한다. 이를 표 1의 표기법으로 나타내면 아래와 같다.

$$\text{분할 효율} = \frac{size_{ngram}}{size_{front} + size_{back}} \quad (1)$$

□

정의 4의 분할 효율은 수식 (1)~(5)와 같이 계산할 수 있다. 여기에서, $SNDO_1O_2$ 릴레이션을 이용하여 n-gram 역색인 혹은 n-gram/2L 역색인에서의 오프셋 개수를 계산한다. $SNDO_1O_2$ 릴레이션은 n-gram 역색인을 제1정규형으로 표현한 릴레이션이므로, 오프셋의 개수가 곧 투플의 개수가 된다. n-gram 역색인을 나타내는 $SNDO_1O_2$ 릴레이션에서 애트리뷰트 S의 값이 s인 투플들의 개수는 k_{ngram} (s) × k_{doc} (s)이다. 이는 보조정리 2에서 보인 바와 같이 $SNDO_1O_2$ 릴레이션에서 애트리뷰트 S의 값에 대해 애트리뷰트 NO_1과 애트리뷰트 DO_2의 값들이 카테시안 곱 형태를 이루기 때문이다. 따라서, n-gram 역색인의 전체 크기는 모든 m-subsequence에 대한 k_{ngram} (s) × k_{doc} (s)의 합계인 수식 (2)가 된

다. 프런트-엔드 역색인에 해당하는 SNO_1 릴레이션에서 애트리뷰트 S의 값이 s인 투플들의 개수는 $k_{ngram}(s)$이다. 따라서, 프런트-엔드 역색인의 전체 크기는 모든 m-subsequence에 대한 $k_{ngram}(s)$의 합계인 수식 (3)이 된다. 백-엔드 역색인에 해당하는 SDO_2 릴레이션에서 애트리뷰트 S의 값이 s인 투플들의 개수는 $k_{doc}(s)$이다. 따라서, 백-엔드 역색인의 전체 크기는 모든 m-subsequence에 대한 $k_{doc}(s)$의 합계인 수식 (4)가 된다. 그 결과로서 수식 (1)~(4)로부터 수식 (5)를 얻는다.

$$size_{ngram} = \sum_{s \in S} (k_{ngram}(s) \times k_{doc}(s)) \quad (2)$$

$$size_{front} = \sum_{s \in S} k_{ngram}(s) \quad (3)$$

$$size_{back} = \sum_{s \in S} k_{doc}(s) \quad (4)$$

$$\text{분할 효율} = \frac{\sum_{s \in S} (k_{ngram}(s) \times k_{doc}(s))}{\sum_{s \in S} (k_{ngram}(s) + k_{doc}(s))} \approx \frac{|S|\ (avg_{ngram}(S) \times avg_{doc}(S))}{|S|\ (avg_{ngram}(S) + avg_{doc}(S))} \quad (5)$$

분할 효율은 문서 집합을 전처리하여 S, $k_{ngram}(s)$, $k_{doc}(s)$를 구함으로써 계산할 수 있다. 이는 문서 집합을 순차적으로 한 번 스캔함으로써 처리할 수 있다(O(*data size*)). 이와같이 m_o의 후보가 될만한 m값들에 대해 분할 효율을 구하고, 그 중 최대 분할 효율을 가지는 m_o를 찾는다. 실험 결과 10MByte부터 1GByte의 크기를 가지는 문서 데이타에 대해서 m_o는 주어진 n-gram의 길이 n보다 1~3정도 긴 $(n+1) \sim (n+3)$의 길이에서 대체적으로 결정되었다.

수식 (5)에서 알 수 있듯이 n-gram 역색인의 공간 복잡도는 $O(|S| \times (avg_{ngram} \times avg_{doc}))$이고, n-gram/2L 역색인의 공간 복잡도는 $O(|S| \times (avg_{ngram} + avg_{doc}))$이다. 수식 (5)에서 분할 효율이 최대가 되는 조건은 주어진 데이타에 대해 avg_{ngram}과 avg_{doc}이 같아지는 경우이다. 그런데, avg_{doc}은 데이타가 커지면 자연히 증가하는 반면, avg_{ngram}은 m이 커져야 증가한다. 따라서, 주어진 데이타를 전처리하여 m_o를 구해보면 데이타가 클 수록 m_o도 더 큰 값으로 결정되는 경향이 있다. 정리하자면, 주어진 데이타가 커질 때 avg_{ngram}과 avg_{doc}도 증가하고, 이때 $(avg_{ngram} + avg_{doc})$보다 $(avg_{ngram} \times avg_{doc})$이 더 많이 증가하므로 분할 효율도 더 커진다. 따라서, n-gram/2L 역색인은 더 큰 데이타에서 색인의 크기를 보다 더 효과적으로 줄여주는 좋은 특성을 가진다.

5.3 질의 처리 성능의 분석

n-gram/2L 역색인의 질의 처리 성능에 영향을 미치는 파라미터는 질의 문자열 Q의 길이인 $Len(Q)$와 m, n이다. 본 절에서는 이들 파라미터의 변화에 따른 n-gram/2L 역색인의 질의 처리 성능의 경향을 알기 위한 개략적인 분석을 한다.

먼저 분석을 단순화하기 위해 다음 두 가지 가정을 한다. 첫째, 역색인에서의 질의 처리 시간은 액세스하는 오프셋들의 개수와 액세스하는 포스팅 리스트의 개수에 비례한다고 가정한다. 포스팅 리스트를 액세스할 때에는 디스크 헤드를 옮기는 시크 타임(seek time) 등이 발생하므로 액세스하는 포스팅 리스트의 개수가 많아질 수록 추가적으로 소요되는 질의 처리 시간이 커진다. 둘째, 알파벳을 Σ라 할때, 문서 데이타가 매우 커서 모든 가지 수($=|\Sigma|^n$)의 n-gram들이 n-gram 역색인 및 프런트-엔드 역색인에 색인되고, 마찬가지로 모든 가지 수($=|\Sigma|^m$)의 m-subsequence들이 백-엔드 역색인에 색인된다고 가정한다(예를 들어 $|\Sigma|=26$, $m=5$이면, $|\Sigma|^m = 11{,}881{,}376$). 질의 처리 성능은 특히 대용량 데이타에서 중요하므로 이러한 가정은 합리적이라고 할 수 있다.

n-gram/2L 역색인의 질의 처리 시간에 대한 n-gram 역색인의 질의 처리 시간의 비율은 수식 (6)~(9)와 같이 계산할 수 있다. 포스팅 리스트당 오프셋들의 평균 개수를 k_{offset}이라 하고 질의 처리를 위해 액세스하는 포스팅 리스트들의 개수를 k_{plist}라 할때, 질의 처리를 위해 액세스하는 오프셋들의 개수는 $k_{offset} \times k_{plist}$이다. n-gram 역색인에서 k_{offset}은 $(size_{ngram} / |\times|^n)$이고 k_{plist}는 $(Len(Q) - n + 1)$이므로, n-gram 역색인의 질의 처리 시간은 수식 (6)이 된다. n-gram/2L 역색인의 프런트-엔드 역색인에서 k_{offset}은 $(size_{front} / |\Sigma|^n)$이고 k_{plist}는 n-gram 역색인에서와 마찬가지로 $(Len(Q) - n + 1)$이므로, 질의 처리 시간은 수식 (7)이 된다. n-gram/2L 역색인의 백-엔드 역색인에서 k_{offset}은 $(size_{back} / |\Sigma|^m)$이고 k_{plist}는 Q를 커버하는 m-subsequence들의 개수인데, 그 개수는 $Len(Q) < m$인 경우와 $Len(Q) \geq m$인 경우가 서로 다르다. $Len(Q) \geq m$인 경우, 그림 8(a)의 $S_{i+1},\ldots,S_{j-1}$의 경우에 해당하는 m-subsequence들의 개수는 $(Len(Q) - m + 1)$이고, S_i 또는 S_j의 경우에 해당하는 m-subsequence들의 개수는 각각 $\sum_{i=0}^{m-n-1} |\Sigma|^{m-n-i}$이다. $Len(Q) < m$인 경우, 그림 8(b)의 S_k의 경우에 해당하는 m-subsequence들의 개수는 $((m - Len(Q) + 1) \times |\Sigma|^{m-Len(Q)})$이고, S_p 또는 S_q의 경우에 해당하는 m-subsequence들의 개수는 각각 $\sum_{i=0}^{Len(Q)-n-1} |\Sigma|^{m-n-i}$이다. 따라서, 백-엔드 역색인에서의 질의 처리 시간은 수식 (8)이 된다. 수식 (9)은 n-gram/2L 역색인의 질의 처리

시간에 대한 n-gram 역색인의 질의 처리 시간의 비율을 나타낸다.

$size_{ngram} = |S| (avg_{ngram} \times avg_{doc})$이므로 수식 (9)로부터 n-gram 역색인의 질의 처리 시간의 복잡도는 $O(|S| (avg_{ngram} \times avg_{doc}))$이고, 마찬가지로 n-gram/2L 역색인의 질의 처리 시간의 복잡도는 $O(|S| (avg_{ngram} + avg_{doc}))$이다. 즉, n-gram 역색인과 n-gram/2L 역색인의 시간 복잡도는 공간 복잡도와 같다. 따라서, n-gram/2L 역색인은 n-gram 역색인에 비해 더 큰 데이타에서 질의 처리 성능을 보다 더 향상시켜주는 좋은 특성을 가진다.

수식 (6)~(9)에서 알 수 있듯이 n-gram 역색인에서는 $Len(Q)$가 길어지는 것에 비례하여 질의 처리 시간이 증가한다. 그러나, n-gram/2L 역색인에서는 $Len(Q)$가 길어져도 질의 처리 시간이 거의 증가하지 않는다. 프런트-엔드 역색인의 경우 $Len(Q)$가 길어지는 것에 비례하여 질의 처리 시간이 증가하나 그 색인의 크기가 매우 작고, 백-엔드 역색인의 경우 Σ^{m-n}이 매우 커서 (예를 들어, $|\Sigma|$=26, m=6, n=3이면, $|\Sigma|^{m-n}$=17,576) $Len(Q)$가 거의 영향을 미치지 못한다. 프런트-엔드 역색인의 크기가 작은 이유는 문서 집합보다 크기가 훨씬 작은 m-subsequence들의 집합에 대해 색인이 구성되기 때문이다(예를 들어, 문서 집합의 크기가 1GByte이고 m=5일때, m-subsequence 집합의 크기는 13~27 MByte). n-gram 역색인은 질의의 길이가 매우 긴 경우 질의 처리 시간도 매우 크다는 것이 문제점으로 지적되므로[16], $Len(Q)$가 길어질 때 질의 처리 성능이 좋은 것은 중요하다.

n-gram/2L 역색인과 n-gram 역색인의 질의 처리 시간에 대한 더 자세한 분석을 하려면 포스팅 리스트를 액세스하는 시간을 고려해야 한다. 한 개의 포스팅 리스트를 액세스하는 시간을 α라 할 때, 추가적으로 소요되는 질의 처리 시간은 k_{plist}에 α를 곱하여 계산한다. 수식 (6)~(8)에 있는 k_{plist} 항을 이용하여 포스팅 리스트를 액세스하는 시간 $plist_time$을 구하면 수식 (10)~(12)가 된다.

수식 (10), (12)에서 알 수 있듯이 $plist_time_{ngram} = plist_time_{front}$이며, 따라서 n-gram/2L 역색인은 포스팅 리스트 액세스 시간에 있어서 n-gram 역색인에 비해 $plist_time_{back}$만큼의 시간이 더 소요된다. 수식 (12)에서 k_{plist}는 $Len(Q) \geq m$의 경우 $(2\sum_{i=0}^{m-n-1} |\Sigma|^{m-n-i})$가 지배적(dominant)이고, $Len(Q) < m$의 경우 $(2\sum_{i=0}^{Len(Q)-n-1} |\Sigma|^{m-n-i})$가 지배적인데, 각각의 값은 m이 증가함에 따라 지수적으로 커진다. 그러므로, m-subsequence의 길이로 m_o 대신 $(m_o - 1)$을 사용하면 색인 크기 측면에서

$$time_{ngram} = \frac{size_{ngram}}{|\Sigma|^{n}} \times (Len(Q) - n + 1) \tag{6}$$

$$time_{front} = \frac{size_{front}}{|\Sigma|^{n}} \times (Len(Q) - n + 1) \tag{7}$$

$$time_{back} = \begin{cases} \dfrac{size_{back}}{|\Sigma|^{m}} \times \left(Len(Q) - m + 1 + 2\sum_{i=0}^{m-n-1} |\Sigma|^{m-n-i} \right), & \text{if } Len(Q) \geq m \\ \dfrac{size_{back}}{|\Sigma|^{m}} \times \left((m - Len(Q) + 1) \times |\Sigma|^{m-Len(Q)} + 2\sum_{i=0}^{Len(Q)-n-1} |\Sigma|^{m-n-i} \right), & \text{if } Len(Q) < m \end{cases} \tag{8}$$

$$\frac{time_{ngram}}{time_{front} + time_{back}} = \begin{cases} \dfrac{size_{ngram} \times (Len(Q) - n + 1)}{(size_{front} \times (Len(Q) - n + 1)) + \left(size_{back} \times \left(\frac{Len(Q) - m + 1}{|\Sigma|^{m-n}} + c \right) \right)}, & \text{if } Len(Q) \geq m \\ \dfrac{size_{ngram} \times (Len(Q) - n + 1)}{(size_{front} \times (Len(Q) - n + 1)) + \left(size_{back} \times \left(\frac{m - Len(Q) + 1}{|\Sigma|^{Len(Q)-n}} + d \right) \right)}, & \text{if } Len(Q) < m \end{cases} \tag{9}$$

where $c = 2\sum_{i=0}^{m-n-1} \left(\frac{1}{|\Sigma|}\right)^{i}$, $d = 2\sum_{i=0}^{Len(Q)-n-1} \left(\frac{1}{|\Sigma|}\right)^{i}$

약간 손해를 보긴 하지만 질의 처리 성능 측면에서 크게 이득을 볼 수 있다. 이에 제6.2절의 색인 크기 및 질의 처리 성능 실험에서도 (m_o - 1)을 사용한다.

6. 성능평가

6.1 실험 데이타 및 실험 환경

본 실험의 목적은 n-gram/2L 역색인의 크기와 질의 처리 성능이 기존의 n-gram 역색인에 비해 우수함을 보이는 것이다. 두 가지 종류의 역색인에 대해 크기의 측정 기준으로는 수식 (13)에 정의된 색인 크기 비율을 사용하며, 질의 처리 성능의 측정 기준으로는 페이지 액세스 횟수와 질의 처리 시간을 사용한다. 색인 크기 비율은 다음과 같이 정의된다.

$$\text{색인 크기 비율} = \frac{\text{n-gram 역색인을 위해 allocate된 page들의 개수}}{\text{n-gram/2L 역색인을 위해 allocate된 page들의 개수}} \quad (13)$$

실험 데이타로는 두 가지 종류의 실제 데이타를 사용한다. 첫 번째 데이타는 정보 검색 분야에서 성능 평가를 위해 주로 많이 사용되는 TREC 데이타의 WSJ, AP, FR 데이타로서, 태그, 공백, 기호, 숫자를 제외하고 10MByte, 100MByte, 1GByte의 크기를 가지는 세 가지의 데이타를 사용한다. 각각의 데이타를 TREC-10M, TREC-100M, TREC-1G라 표기한다. 두 번째 데이타는 바이오인포메틱스 분야에서 널리 사용되는 NCBI BLAST 웹 사이트[1]의 nr, env_nr, month.aa, pataa 단백질 sequence 데이타로서, 10MByte, 100MByte, 1GByte의 크기를 가지는 세 가지의 데이타를 사용한다. 각각의 데이타를 PROTEIN-10M, PROTEIN-100M, PROTEIN-1G라 표기한다. TREC 데이타의 경우 원래의 데이타로부터 공백, 기호, 숫자를 제거한 이유는 PROTEIN 데이타와 데이타의 포맷을 통일함으로써 데이타 포맷의 차이가 실험 결과에 영향을 미치는 것을 방지하기 위해서이다.

색인 크기를 비교하기 위해, PROTEIN-10M, PROTEIN-100M, PROTEIN-1G와 TREC-10M, TREC-100M, TREC-1G에 대해 m을 변화시키면서 색인 크기 비율의 추정치를 계산하고 실제치를 측정한다. 색인 크기 비율의 추정치로는 제4.2절에서 제시한 분할 효율 (수식 (5))를 사용하여 계산한다. 분할 효율에 의해 구해진 m_o가 실제 최적 길이와 정확히 일치함을 보인다. n-gram 역색인과 프런트-엔드 역색인의 구성에 사용되는 n-gram의 길이는 n-gram의 응용들[2,17]에서 주로 사용하는 3으로 한다. 백-엔드 역색인의 구성에 사용되는 m-subsequence의 길이는 m의 최소값(즉, n+1)인 4부터 (m_o+1)까지 변화시키며 색인 크기를 측정한다.

질의 처리 성능을 비교하기 위해 데이타의 크기와 질의 문자열의 길이를 변화시키면서 두 색인 구조의 질의 처리 시간을 측정한다. 먼저 PROTEIN 데이타와 TREC 데이타의 크기를 10MByte, 100MByte, 1GByte로 변화시키면서 각 데이타에 대해 구성된 두 색인 구조의 질의 처리 시간을 측정한다. 이때, 데이타로부터 3~18의 범위에서 100개의 질의 문자열을 랜덤하게 선택하여 측정하며, 결과로서 그 평균값을 보인다. 다음으로, 질의 문자열의 길이를 3, 6, 9, 12, 15 및 18로 변화시키면서 PROTEIN-1G과 TREC-1G에 대해 구성된 두 색인 구조의 페이지 액세스 횟수와 질의 처리 시간을 측정한다. 두 색인 구조에서 n-gram의 길이가 3이므로 최소 질의 문자열의 길이는 3이다. 각 질의 문자열의 길에에 대해 데이타로부터 랜덤하게 50개씩의 질의 문자열을 선택하여 측정하며, 결과로서 그 평균값을 보인다.

실험은 1GB 메모리를 가진 Pentinum 2.6 MHz Linux PC와 400 GB Segate E-IDE 디스크를 사용하여 수행한다. 리눅스 시스템 자체의 버퍼링 효과를 피하고 실제디스크 I/O를 보장하기 위하여 데이타 및 색인 파일에 대해 원시 디스크(raw disk)를 사용한다. 구현을 위해 역색인이 밀겁합되어 있는 Odysseus ORDBMS [10]를 사용하며, 데이타 및 색인 페이지의 크기는 4,096 바이트로 한다.

6.2 실험 결과

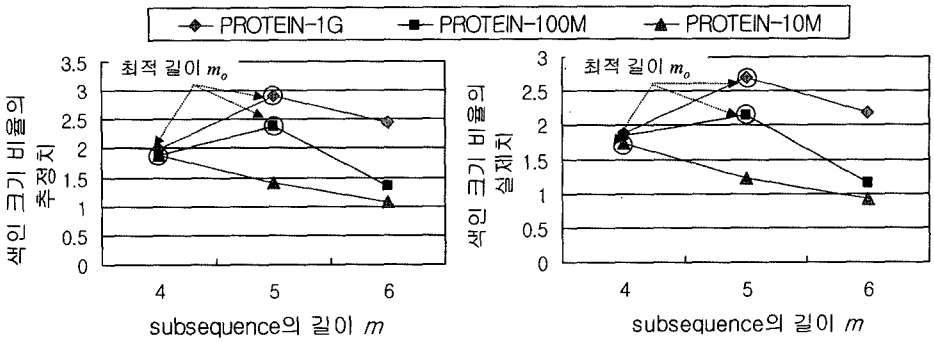
6.2.1 색인 크기

그림 13은 PROTEIN 데이타 크기와 m-subsequence의 길이의 변화에 따른 색인 크기 비율의 추정치와 실제치를 나타낸다. n-gram/2L 역색인의 크기는 n-gram 역색인의 크기에 비해 현저히 줄어들었음을 알 수 있다. 그림 13(b)의 색인 크기 비율의 실제치에서 최적 길이 m_o의 m-subsequence로 구성된 n-gram/2L 역색인은 n-gram 역색인에 비해 색인의 크기를 PROTEIN-10M에서 1.7배 줄였고, PROTEIN-100M에서 2.2배 줄였으며, PROTEIN-1G에서 2.7배 줄였다.

그림 13(a)에서 m_o의 추정치는 그림 13(b)에서 m_o의 실제치와 일치한다. PROTEIN-10M, PROTEIN-100M, PROTEIN-1G에 대해 최적 길이 m_o를 살펴보면 추정치와 실제치 모두 각각 4, 5, 5로 일치하며, 색인 크기 비율의 실제치가 추정치의 86%~98%에 달해 적은 오차(2%~14%)를 보인다. 이는 제5.2절의 분석이 정확함을 보여주는 결과이다.

그림 14는 TREC 데이타의 크기와 m-subsequence의 길이의 변화에 따른 색인 크기 비율의 추정치와 실제치를 나타낸다. 여기에서도 m_o에 대해 추정치와 실제

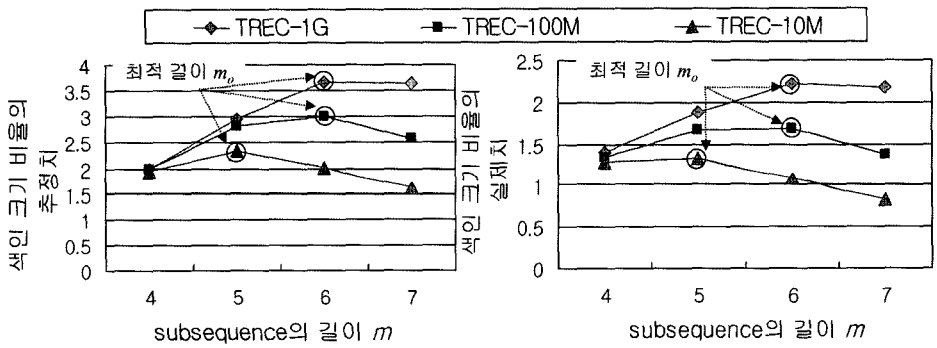
1) http://www.ncbi.nlm.nih.gov/BLAST



(a) m값의 변화에 따른 색인 크기 비율의 추정치.

(b) m값의 변화에 따른 색인 크기 비율의 실제치.

그림 13 PROTEIN 데이타에 대한 색인 크기 비율의 추정치와 실제치



(a) m값의 변화에 따른 색인 크기 비율의 추정치.

(b) m값의 변화에 따른 색인 크기 비율의 실제치.

그림 14 TREC 데이타에 대한 색인 크기 비율의 추정치와 실제치

치가 일치한다. 그러나, 색인 크기 비율의 실제치가 추정치의 52%~70% 밖에 안되어 그림 13에서보다 큰 차이를 보인다. 이는 실제역색인의 구현에서는 오프셋 이외에 문서 식별자나 m-subsequence 식별자에 대한 색인 공간이 소요되기 때문이다. 역색인에서는 용어가 나타난 문서의 식별자(또는 m-subsequence 식별자)와 그 문서 내에서의 오프셋 집합이 저장되며, 이러한 단위를 **포스팅**이라 부른다. 이때, 각각의 포스팅에 저장된 평균 오프셋의 개수가 n-gram 역색인에서보다 n-gram/2L 역색인에서 더 작기 때문에 식별자에 소요되는 색인 공간의 크기가 n-gram 역색인에서보다 n-gram/2L 역색인에서 상대적으로 더 크고, 따라서 실제치와 추정치가 오차를 보였다. 포스팅에 저장된 평균 오프셋 개수에 있어서 두 역색인이 차이를 보이는 이유는 n-gram 역색인은 위치 정보의 중복이 있어 n-gram이 문서에 나타나는 횟수가 많지만, n-gram/2L 역색인은 위치 정보의 중복이 제거되어 n-gram이 m-subsequence에 나타나는 횟수나 m-subsequence가 문서에 나타나는 횟수가 작기 때문이다. 그림 13보다 그림 14에서 이러한 현상이 더 두드러진 이유는 TREC 데이타가 잡지 혹은 신문 기사의 모음이므로 동일한 표현이 여러번 쓰일 수 있다는 특성을 가지고 있기 때문이다.

표 2는 데이타의 크기가 커질 수록 색인 크기 비율이 증가함을 나타낸다. 이때, 색인 크기 비율을 최대로 만드는 m-subsequence의 길이인 m_o에 대한 결과와 질의 처리 성능을 최적화하는 m_o-1에 대한 결과를 보여준다. 실험 결과는 제5.2절의 분석이 정확함을 보여준다. 표 2에서 $m = m_o$이고 데이타의 크기가 10MB, 100MB, 1GB로 10배씩 늘어날 때, PROTEIN 데이타에서의 색인 크기 비율은 평균 25%씩 증가하고, TREC 데이타에서의 색인 크기 비율은 평균 29%씩 증가한다. 마찬가지로 $m = m_o$-1일 때, PROTEIN 데이타에서의 색인 크기 비율은 평균 2%씩 증가하고, TREC 데이타에서의 색인 크기 비율은 평균 21%씩 증가한다. 이때, m은 n

표 2 데이타 크기의 증가에 따른 색인 크기 비율

데이타 종류	10 MByte	100 MByte	1 GByte
PROTEIN	1.734 (m_o=4)	2.153 (m_o=5)	2.705 (m_o=5)
	N/A (m_o-1=3)	1.847 (m_o-1=4)	1.877 (m_o-1=4)
TREC	1.337 (m_o=5)	1.678 (m_o=6)	2.219 (m_o=6)
	1.281 (m_o-1=4)	1.677 (m_o-1=5)	1.878 (m_o-1=5)

보다 커야하기 때문에 PROTEIN-10M에서 m = m_o-1 = 3에 대한 실험 결과는 없다.

6.2.2 질의 처리 성능

그림 15(a)는 PROTEIN 데이타의 크기가 10MByte, 100MByte, 1GByte로 증가할 때 각각의 데이타에 대해 구성된 n-gram 역색인과 n-gram/2L 역색인에서의 질의 처리 시간을 나타낸다. 이때, m-subsequence의 길이로 (m_o-1)을 사용한 실험 결과를 보인다. 제5.3절에서 분석한 바와같이 n-gram/2L 역색인은 데이타의 크기가 커질수록 n-gram 역색인에 비해 상대적으로 질의 처리 성능이 더 좋아진다. 그림 15(a)에서 데이타의 크기가 100MByte에서 1GByte로 증가할 때, n-gram 역색인에 비해 n-gram/2L 역색인의 질의 처리 성능이 향상되는 비율은 1.37배에서 6.65배로 증가하였다.

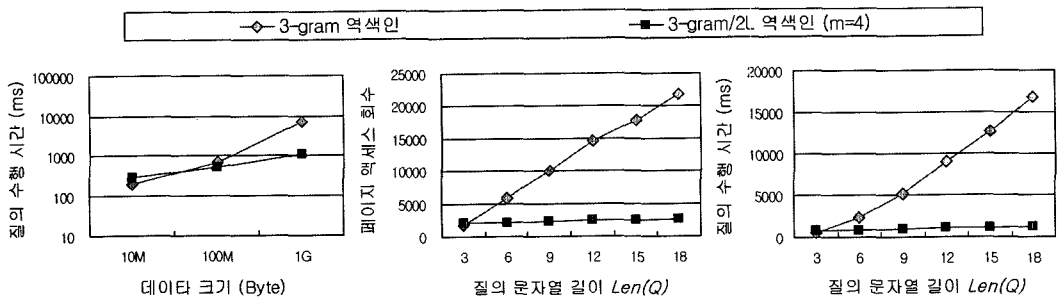
그림 15(b)와 (c)는 PROTEIN-1G에 대해 구성된 두 가지 역색인에서 질의 문자열의 길이가 변화할 때의 페이지 액세스 횟수와 질의 처리 시간을 나타낸다. $Len(Q)$가 증가함에 따라 n-gram 역색인에서의 페이지 액세스 횟수와 질의 처리 시간은 상대적으로 가파르게 증가하는데 반해, n-gram/2L 역색인에서는 $Len(Q)$에 거의 영향을 받지않고 아주 조금씩만 증가한다. 그림 15(b)에서 $Len(Q)$가 3에서 18로 증가함에 따라 n-gram 역색인의 페이지 액세스 횟수는 12.0배, 질의 처리 시간은 32.9배 증가한 반면, n-gram/2L 역색인의 페이지 액세스 횟수는 27%, 질의 처리 시간은 53% 증가한다. 결과적으로 질의 처리 시간에 있어서 n-gram/2L 역색인은 n-gram 역색인에 비해 3~18 범위의 길이를 가지는 질의들에 대해 평균 13.1배의 성능 향상을 보였다.

그림 16은 TREC 데이타에 대한 질의 처리 성능의 결과이며, 그림 15에서와 유사한 결과를 보여준다.

7. 결 론

본 논문에서는 기존 n-gram 역색인에 비해 그 크기를 줄이고 질의 처리 성능을 향상시킨 n-gram/2L 역색인을 제안하였다. 제안하는 색인 구조에 사용된 핵심 기법은 n-gram 역색인에 존재하는 위치 정보의 중복을 발견하고, 그 중복을 제거한 것이다. 위치 정보의 중복을 제거하기 위해 사용한 방법은 문서로부터 n-gram들을 직접 추출하여 색인하는 것 대신에, 문서로부터 길이가 m인 m-subsequence들을 n-1씩 겹치도록 추출하여 백-엔드 역색인으로 색인하고, 다시 m-subsequence들로부터 n-gram들을 추출하여 프런트-엔드 역색인으로 색인하는 것이다.

본 논문에서는 n-gram/2L 역색인이 가지는 중요한 특성들에 대해 이론적으로 분석하였다. 먼저, n-gram 역색인에 존재하는 위치 정보의 중복이 관계형 데이타베이스 이론에서의 의미 있는 다치 종속성과 같은 것임을 보조정리 2에서 증명하였다. 그리고, 제1정규형으로 표현된 n-gram 역색인을 제4정규형으로 정규화함으로

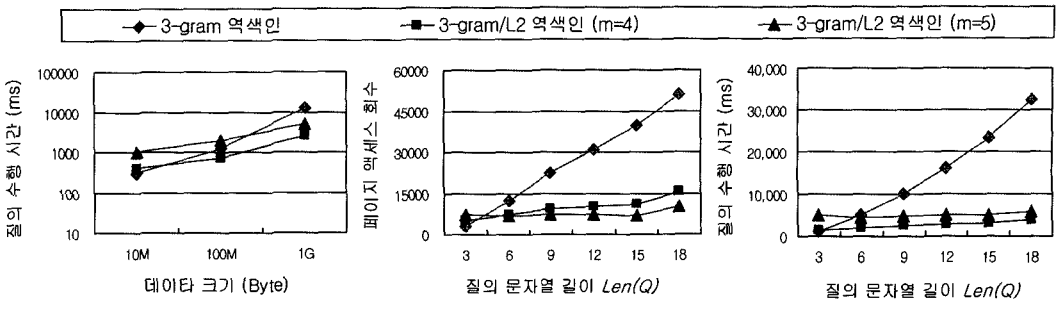


(a) 데이타 크기의 변화에 따른 질의 처리 시간. (*Len(Q)*: 3~18)

(b) *Len(Q)*의 변화에 따른 page 액세스 횟수. (데이타: PROTEIN-1G)

(c) *Len(Q)*의 변화에 따른 질의 처리 시간. (데이타: PROTEIN-1G)

그림 15 PROTEIN 데이타에 대한 질의 처리 성능 결과



(a) 데이타 크기의 변화에 따른 질의 처리 시간. (*Len(Q)*: 3~18)

(b) *Len(Q)*의 변화에 따른 page 액세스 횟수. (데이타: TREC-1G)

(c) *Len(Q)*의 변화에 따른 질의 처리 시간. (데이타: TREC-1G)

그림 16 TREC 데이타에 대한 질의 처리 성능 결과

써 n-gram/2L 역색인이 구해짐을 정리 2에서 증명하였다. 다음으로, 제5.2절에서 n-gram/2L 역색인의 크기를 분석하였으며, 그 크기를 최소로 만드는 m의 값(즉 m_o)을 결정하기 위한 모델을 제시했다. 또한, n-gram/2L 역색인의 공간 복잡도는 $O(|S| (avg_{ngram} + avg_{doc}))$으로서 n-gram 역색인의 공간 복잡도인 $O(|S| (avg_{ngram} \times avg_{doc}))$에 비해 대용량 데이타베이스에서 색인의 크기가 줄어드는 비율이 더 커짐을 보였다. 다음으로, 제5.3절에서 n-gram/2L 역색인의 질의 처리 성능을 분석하였으며, 색인 크기 측면에서 m_o보다 약간 손해를 보긴 하지만 질의 처리 성능 측면에서의 큰 이득을 얻기 위해서 m의 값으로 (m_o-1)이 좋음을 제시했다. 또한, n-gram/2L 역색인의 시간 복잡도는 $O(|S| (avg_{ngram} + avg_{doc}))$으로서, n-gram 역색인의 시간 복잡도인 $O(|S| (avg_{ngram} \times avg_{doc}))$에 비해 대용량 데이타베이스에서 질의 처리 성능이 좋아지는 비율이 더 커짐을 보였다. 마지막으로, 질의의 길이가 길어져도 질의 처리 시간이 매우 약간씩만 증가함을 분석적으로 보였다(수식 (9)).

본 논문에서는 데이타의 종류, m-subsequence의 길이, 데이타의 크기, 질의 문자열의 길이를 달리하면서 색인의 크기와 질의 처리 성능에 대해 많은 실험을 수행하였다. 실제 1GByte의 텍스트 데이타와 단백질 sequence 데이타에 대한 실험을 통하여, m-subsequence의 길이로 (m_o-1)을 사용했을 때, n-gram/2L 역색인은 n-gram 역색인에 비해 최대 1.9배(PROTEIN-1G, m=4)~2.7배(PROTEIN-1G, m=5) 더 작은 크기를 가지면서, 동시에 3~18 범위의 길이를 가지는 질의들에 대해 질의 처리 성능을 최대 13.1배(PROTEIN-1G, m=4) 향상시킴을 보였다.

이 같은 결과로 볼 때, n-gram/2L 역색인은 기존 n-gram 역색인을 대체할 수 있는 새로운 색인 구조라고 사료된다. n-gram/2L 역색인은 최근 그 중요성이 부각되고 있는 단백질과 DNA sequence에 대한 유사 sequence 매칭 등의 질의도 효율적으로 처리할 수 있을 것으로 예상되며, 이 부분에 대한 자세한 내용은 향후 연구로서 수행한다.

참 고 문 헌

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto, *Modern Information Retrieval*, ACM Press, 1999.

[2] Hugh E. Williams and Justin Zobel, "Indexing and Retrieval for Genomic Databases," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 14, No. 1, pp. 63-78, Jan./Feb. 2002.

[3] Gonzalo Navarro, "A Guided Tour to Approximate String Matching," *ACM Computing Surveys*, Vol. 33, No. 1, pp. 31-88, Mar. 2001.

[4] I. Witten, A. Moffat, and T. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, Morgan Kaufmann Publishers, Los Altos, California, 2nd ed., 1999.

[5] James Mayfield and Paul McNamee, "Single N-gram Stemming," In *Proc. Int'l Conf. on Information Retrieval*, ACM SIGIR, Toronto, Canada, pp. 415-416, July/Aug. 2003.

[6] Ethan Miller, Dan Shen, Junli Liu, and Charles Nicholas, "Performance and Scalability of a Large-Scale N-gram Based Information Retrieval System," Journal of Digital Information 1(5), pp. 1-25, Jan. 2000.

[7] Alistair Moffat and Justin Zobel, "Self-indexing inverted files for fast text retrieval," *ACM Trans. on Information Systems*, Vol. 14, No. 4, pp. 349-379, Oct. 1996.

[8] Falk Scholer, Hugh E. Williams, John Yiannis and Justin Zobel, "Compression of Inverted Indexes for Fast Query Evaluation," In *Proc. Int'l Conf. on Information Retrieval*, ACM SIGIR, Tampere, Finland, pp. 222-229, Aug. 2002.

[9] Joon Ho Lee and Jeong Soo Ahn, "Using n-Grams for Korean Text Retrieval," In *Proc. Int'l Conf. on Information Retrieval*, ACM SIGIR, Zurich, Switzerland, pp. 216-224, 1996.
[10] Kyu-Young Whang, Min-Jae Lee, Jae-Gil Lee, Min-soo Kim, and Wook-Shin Han, "Odysseus: a High-Performance ORDBMS Tightly-Coupled with IR Features," In *Proc. the 21th IEEE Int'l Conf. on Data Engineering(ICDE)*, Tokyo, Japan, Apr. 2005.
[11] Jonathan D. Cohen, "Recursive Hashing Functions for n-Grams," *ACM Trans. on Information Systems*, Vol. 15, No. 3, pp. 291-320, July 1997.
[12] Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley, 4th ed., 2003.
[13] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan, *Database Systems Concepts*, McGraw-Hill, 4th ed., 2001.
[14] Jeffery D. Ullman, *Principles of Database and Knowledge-Base Systems Vol. I*, Computer Science Press, USA, 1988.
[15] Raghu Ramakrishnan, *Database Management Systems*, McGraw-Hill, 1998.
[16] Hugh E. Williams, "Genomic Information Retrieval," In *Proc. the 14th Australasian Database Conferences*, 2003.
[17] Karen Kukich, "Techniques for Automatically Correcting Words in Text," *ACM Computing Surveys*, Vol. 24, No. 4, pp. 377-439, Dec. 1992.

부 록

A. 정리 1의 증명

문서 d가 문자의 sequence $c_0,c_1,...,c_{N-1}$로 주어졌을 때, 문서 d로부터 추출되는 n-gram들의 개수는 $N-n+1$개이고 각 n-gram의 시작 문자는 c_i이다($0 \le i < N-n+1$). 또한, subsequence의 길이가 m으로 주어졌을 때, 문서 d로부터 추출되는 subsequence들의 개수는 $\lfloor (N-n+1) / (m-n+1) \rfloor$개이고 각 subsequence의 시작 문자는 c_j이다($0 \le j < \lfloor (N-n+1) / (m-n+1) \rfloor$). 설명의 편의상 c_i를 시작 문자로 하는 n-gram을 N_i로 표기하고, c_j를 시작 문자로 하는 subsequence를 S_j로 표기한다. 문서 d로부터 추출되는 인접한 임의의 두 subsequence를 S_p, S_q라 할 때, S_p 로부터 추출되는 n-gram들은 $N_p,...,N_{p+m-n}$이고, S_q로부터 추출되는 n-gram들은 $N_q,...,N_{q+m-n}$이다. Subsequence들끼리 서로 겹치는 구간의 길이를 l이라 할 때, 길이 l은 $l = n-1$, $l < n-1$, $l > n-1$의 세 가지 경우로 나뉘는데, 각각의 경우들에 대해 두 subsequence S_p, S_q로부터 추출되는 n-gram들 중 누락되거나 중복되는 n-gram이 있는지를 검사함으로써 정리를 증명한다.

- Case $l = n-1$: 서로 $n-1$만큼씩 겹치므로 $q=p+m-n+1$이고, S_q로부터 추출되는 n-gram들은 다시 $N_{p+m-n+1},...,N_{p+2m-2n+1}$로 표현된다. S_p로부터 추출되는 n-gram들은 $N_p,...,N_{p+m-n}$이므로, 두 subsequence S_p, S_q로부터 $N_p,...,N_{p+m-n},N_{p+m-n+1},...,N_{p+2m-2n+1}$의 n-gram들이 누락되거나 중복되지 않고 한 번씩만 추출된다.
- Case $l < n-1$: $l = n-2$이라고 가정하자. 서로 $n-2$만큼씩 겹치므로 $q=p+m-n+2$이고, 따라서 S_q로부터 추출되는 n-gram들은 다시 $N_{p+m-n+2},...,N_{p+2m-2n+2}$로 표현된다. 두 subsequence S_p, S_q로부터 n-gram $N_{p+m-n+1}$이 한 번도 추출되지 않으므로, 누락되는 n-gram이 발생한다.
- Case $l > n-1$: $l = n$이라고 가정하자. 서로 n만큼씩 겹치므로 $q=p+m-n$이고, 따라서 S_q로부터 추출되는 n-gram들은 다시 $N_{p+m-n},...,N_{p+2m-2n}$로 표현된다. 두 subsequence S_p, S_q로부터 n-gram N_{p+m-n}이 각각 한 번씩, 총 두 번 추출되므로, 중복되는 n-gram이 발생한다.

따라서, subsequence 추출 과정에서 인접한 두 subsequence들이 서로 $n-1$만큼씩 겹치도록 해야만 n-gram들의 누락이나 중복을 방지할 수 있다.

B. 보조정리 3의 증명

SNO_1 릴레이션에 있는 다치 종속성은 $SO_1 \rightarrow\rightarrow N$와 $SNO_1 \rightarrow\rightarrow S|N|O_1$이다. 또한, SDO_2 릴레이션에 있는 다치 종속성은 $DO_2 \rightarrow\rightarrow S$와 $SDO_2 \rightarrow\rightarrow S|D|O_2$이다. 이 다치 종속성들은 모두 무의미 다치 종속성이므로 제4정규형을 위배하지 않는다.

김 민 수

2000년 3월~현재 한국과학기술원 전자전산학과 전산학전공 박사. 1998년 3월~2000년 2월 한국과학기술원 전산학과 석사. 1994년 3월~1998년 2월 한국과학기술원 전산학과 학사. 관심 분야는 객체관계형 데이타베이스 시스템, 정보 검색, XML 데이타베이스

황 규 영

1973년 서울대학교 전자공학과 졸업(B.S). 1975년 한국과학기술원 전기 및 전자학과 졸업(M.S.). 1982년 Stanford University(M.S.). 1983년 Stanford University(Ph.D.). 1975년~1978년 국방과학연구소(ADD), 선임연구원. 1983년~1990년 IBM T.J. Watson Research Center, Research Staff Member. 1992년~1994년 한국정보과학회 데이타베이스 연

구회(SIGDB) 운영위원장. 1995년 한국정보과학회 이사 겸 논문지 편집위원장. 1999년~2000년 한국정보과학회 부회장. Editor-in-Chief: The VLDB Journal, 2003년~현재. Editor: IEEE Transactions on Knowledge and Data Engineering, 2002년~현재. Editor: The VLDB Journal, 1990년~2003년. Editor: Distributed and Parallel Databases: An International Journal, 1991년~1995년. Editor: International Journal of Geographical Information Science, 1994년~현재. Associate Editor: The IEEE Data Engineering Bulletin, 1990년~1993년. 1998년~2004년 Trustee, The VLDB Endowment. 1999년~2005년 Steering Committee Member, DASFAA. 1999년~현재 한국과학기술원 전자전산학과 전산학전공 교수. 1990년~현재 첨단정보기술연구센터(과학재단 우수연구센터) 소장. 관심분야는 데이타베이스 시스템, 멀티미디어, GIS

이 재 길

2005년 3월~현재 한국과학기술원 BK21 박사후연구원. 1999년 3월~2005년 2월 한국과학기술원 전자전산학과 전산학전공 박사. 1997년 3월~1999년 2월 한국과학기술원 전산학과 석사. 1993년 3월~1997년 2월 한국과학기술원 전산학과 학사. 2001년 7월~2001년 8월 Visiting Scholar, 미국 HP Labs. 관심 분야는 객체 관계형 데이타베이스 시스템, 정보 검색, 질의 최적화, XML 데이타베이스, 데이타베이스 보안

이 민 재

2004년 12월~현재 (주)네오위즈 연구원 2004년 3월~2004년 11월 박사후연구원 첨단정보기술연구센터, KAIST. 2004년 2월 한국과학기술원 전자전산학과 전산학전공 박사. 1997년 2월 한국과학기술원 전자전산학과 전산학전공 석사. 1995년 2월 한국과학기술원 전자전산학과 전산학전공 학사. 관심분야는 지리 정보 시스템, 객체 관계형 데이타베이스 시스템