비트맵 필터를 이용한 효율적인 유사 문자열 검색 기법1)

권인택*, 김종익**
*전북대학교 전자정보공학부
**전북대학교 컴퓨터공학부
e-mail: {*kit0123, **jongik}@jbnu.ac.kr

Efficient Approximate String Searches using Bitmap Filter

Inteak Kwon*, Jongik Kim**
*Div. of Electronics & Information Engineering, Chonbuk National University
**Div. of Computer Science & Engineering, Chonbuk National University

요 약

텍스트 데이터는 표현 방식의 차이, 타이핑 오류 등을 포함하고 있어 정확히 일치하는 검색으로는 유용한 정보를 얻기 어렵다. 따라서 유사도 기반 검색 방법이 많이 연구되고 있으며 효율적인 유사도 기반 검색을 위해 텍스트 데이터에 대한 역 리스트를 구성한다. 그리고 이를 병합하여 질의와 일정 기준 이상 유사한 데이터를 찾는다. 본 논문에서는 역 리스트 병합 과정에서 역 리스트의 탐색 비용을 줄이기 위해 비트맵 필터를 사용하는 기법을 제안한다. 비트맵 필터를 사용하여 역 리스트의 탐색 여부를 결정하여 불필요한 역 리스트 탐색을 회피함으로써 역 리스트 병합 비용을 줄인다. 실험을 통하여 제안된 기법이 기존의 연구에서 제안된 역 리스트 병합 알고리즘의 성능을 30~40% 정도 개선함을 보인다.

1. 서론

인터넷 상에 존재하는 대부분의 정보는 텍스트 형태로 표현되어 있다. 텍스트 데이터는 표현방식의 차이, 타이핑 오류 등을 포함하고 있어 정확히 일치하는 검색만을 수행하는 경우 유용한 정보를 얻어내기가 매우 어렵다. 따라서 텍스트 데이터로부터 원하는 정보를 추출하기 위해 유사도를 기반으로 하는 검색 방법이 많이 연구되고 있다.

유사도를 기반으로 하는 텍스트 검색은 주어진 문자열 집합 내에서 질의 문자열과 유사한 모든 문자열을 찾아내는 문제로 정의할 수 있다. 유사 문자열 검색을 위해서는 서로 다른 두 문자열의 유사도를 측정할 수 있어야 한다. 문자열의 유사도를 측정하기 위해 편집거리, 코사인 유사도, 자카드 유사도와 같은 다양한 유사도 기준이 제안되었고 기존의 연구에서는 이들을 문자열을 집합으로 변환한 후 유사도를 두 집합의 교집합의 크기로 변환하여 유사도로 사용한다[1]. 문자열을 집합으로 변환하기 위해 문자열을 특정 길이의 부분 문자열로 분해하는 방법이 가장 많이 사용된다. 예를 들어 “conference”라는 문자열을 길이 4인 부분 문자열로 분해하면 {conf, onfe, fere, eren, ence}의 부분 문자열 집합을 얻을 수 있다. 이때, 길이 q인 부분 문자열을 q-그램이라 부른다.

1) "이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2010-0005466)"

유사 문자열 검색은 주어진 문자열 집합 내의 각 문자열과 질의 문자열을 q-그램 집합으로 변환한 후에 각 문자열과 질의 문자열 사이의 교집합의 크기를 직접 계산함으로써 수행할 수 있다. 하지만, 주어진 문자열 집합이 매우 큰 경우 이러한 방법은 많은 연산을 필요로 하여 매우 비효율적이다.

효율적인 유사 문자열 검색을 위해 문자열 집합의 각 문자열들을 q-그램으로 분해하고 각 q-그램들에 대해 자신을 포함하는 모든 문자열들의 ID를 정렬된 리스트로 나타내게 한다. 이때, 이 리스트를 역 리스트라고 한다. 그리고 질의 처리 시 질의 문자열을 분해하여 q-그램들을 생성하고 이 q-그램들에 해당하는 역 리스트들을 병합한다. 병합된 리스트에서 문자열 ID가 나타난 횟수는 질의 문자열과 문자열 집합의 각 문자열들의 교집합의 크기가 된다. 따라서 역 리스트들을 병합함으로써 질의 문자열과의 교집합의 크기가 일정 이상인 문자열들을 효율적으로 찾을 수 있다.

본 논문에서는 역 리스트의 병합 과정에서 비트맵 필터를 이용하여 역 리스트 탐색 비용을 줄이는 기법을 제안한다. 제안하는 기법은 역 리스트를 탐색하기 전에 비트맵 필터를 확인하여 역 리스트의 탐색 여부를 결정하고 이를 통해 불필요한 역 리스트 탐색을 회피하여 역 리스트의 병합 비용을 절감한다.

본 논문의 구성은 다음과 같다. 2장에서는 대표적인 리스트 병합 알고리즘인 MergeOpt[1]에 대해 설명하며 3장

에서는 제안하는 비트맵 필터와 이를 이용한 역 리스트 병합 방법에 대해 알아본다. 그리고 4장에서는 실험을 통하여 제안하는 기법의 성능을 평가하며 5장에서 결론을 맺는다.

2. 대표적인 리스트 병합 알고리즘

역 리스트를 기반으로 하는 유사 문자열 검색은 문자열 집합의 문자열들을 역 리스트들로 나타내고 이를 병합함으로써 질의 문자열과의 교집합의 크기를 계산할 수 있으며 이와 같은 방식으로 교집합의 크기가 최소 T이상인 문자열들을 찾는다. 본 장에서는 대표적인 리스트 병합 알고리즘인 MergeOpt에 대해 설명한다.

MergeOpt는 역 리스트들을 길이에 따라 T-1개의 긴 역 리스트들과 나머지인 짧은 역 리스트들로 분류하며 이들을 각각 Long List와 Short List라고 한다. 그리고 Short List에서 나타나지 않는 문자열 ID는 질의 결과가 될 수 없다는 점을 이용하여 Short List에 나타난 문자열 ID들로 Long List로 분류된 역 리스트들을 이진 탐색한다.

먼저 Short List로 분류된 역 리스트들을 병합하여 하나의 정렬된 리스트를 생성한다. 이 리스트에 포함된 각 문자열 ID들로 Long List로 분류된 역 리스트들을 이진 탐색하여 질의 문자열에 의해 추출된 역 리스트들에서 T번 이상 나타나는지 확인한다.

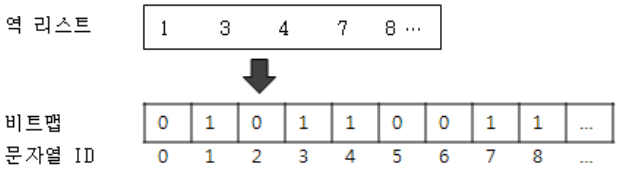
3. 비트맵 필터를 이용한 역 리스트 탐색 기법

본 장에서는 제안하는 비트맵 필터를 이용한 역 리스트 탐색 기법에 대해 설명한다. 제안하는 기법은 역 리스트를 탐색하기 전에 비트맵 필터를 이용하여 문자열 ID가 포함되어 있을 가능성이 있는 역 리스트의 개수를 파악한다. 그리고 이를 이용하여 불필요한 역 리스트 탐색을 회피하는 방법으로 리스트 병합 비용을 절감한다.

3.1. 역 리스트의 비트맵

MergeOpt는 Long List로 분류된 역 리스트들이 문자열 ID를 포함하고 있는지 확인하기 위하여 이진 탐색을 이용한다. (그림 1)과 같은 역 리스트에 대한 비트맵을 구성한다고 가정해보자. 비트맵의 길이는 문자열 집합의 크기와 같고 각 비트는 해당하는 문자열 ID가 역 리스트에 포함되어 있는지를 나타낸다. 따라서 역 리스트에 문자열 ID 1, 3, 4, 8이 포함되어 있으므로 비트맵의 1, 3, 4, 8번째 비트만 1로 표시한다. 이렇게 구성된 비트맵을 이용하면 문자열 ID가 역 리스트에 포함되어 있는지 상수 시간에 확인할 수 있다. 따라서 이진 탐색을 통하여 문자열 ID가 역 리스트에 포함되어 있는지 확인하는 것보다 더 효율적이다.

이와 같은 비트맵을 이용하여 질의를 처리하기 위해서는 모든 역 리스트에 대해 (그림 1)과 같은 비트맵을 구성해야 한다. 하지만 문자열 집합이 큰 경우 구성되는 역 리스트의 개수가 많아지고 이러한 모든 역 리스트들은 문자열 집합 크기의 비트맵이 필요하다. 따라서 많은 공간 비용이 요구되기 때문에 비효율적이다.

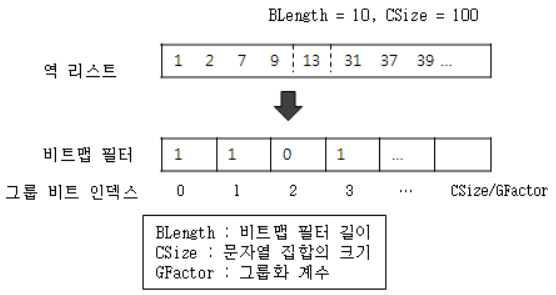


(그림 1) 역 리스트의 비트맵 표현

3.2. 비트맵 필터의 구성

본 절에서는 비트맵의 공간 비용을 줄이기 위해 비트맵의 길이를 고정하고 문자열 ID를 비트맵 길이만큼의 그룹으로 나누어 각 그룹에 하나의 비트만 할당하는 비트맵 필터에 대해 설명한다.

비트맵 필터의 길이를 BLength라고 하고 문자열 집합의 크기를 CSize라고 하자. 이때 문자열 ID들은 BLength 만큼의 그룹으로 나누어지고 한 그룹으로 나누어지는 문자열 ID의 수를 그룹화 계수 GFactor라 하고 CSize/BLength로 계산된다. 따라서 그룹화 계수만큼의 문자열 ID들은 역 리스트에 포함 여부를 나타내기 위해 하나의 비트를 공유하게 된다. 이때 공유되는 비트를 그룹 비트라고 부른다. 각 그룹 비트는 그룹 비트를 공유하는 문자열 ID들 중 한 개 이상이 역 리스트에 포함되어 있는 경우 그룹 비트는 1로 표시되며 그렇지 않은 경우는 0으로 표시된다.



(그림 2) 역 리스트에 대한 비트맵 필터

(그림 2)와 같은 경우 CSize가 100이고 BLength가 10이므로 GFactor는 10이 된다. 따라서 10개의 문자열 ID들이 하나의 비트를 공유하게 된다. (그림 2)처럼 비트맵 필터의 0번째 비트는 문자열 ID 0부터 9까지의 포함 여부를 나타내며 1,2,7,9가 포함되어 있으므로 1로 표시된다. 하지만 2번째 비트의 경우 문자열 ID 20부터 29까지의 포함 여부를 나타내지만 해당하는 문자열 ID들이 모두 역 리스트에 포함되어 있지 않으므로 0으로 표시된다. (그림 3)은 앞서 설명한 비트맵 필터의 구성 방법을 기술한 알고리즘이다.

문자열 집합의 크기가 고정되어 있으므로 그룹화 계수

는 오직 비트맵 필터의 길이의 영향만 받는다. 비트맵 필터의 길이가 작으면 그룹화 계수는 커지게 되고 하나의 그룹 비트를 공유하는 문자열 ID가 많아진다. 하나의 그룹 비트를 공유하는 문자열 ID가 많아지면 불필요한 역 리스트 탐색을 효율적으로 회피 할 수 없다. 이러한 이유로 비트맵 필터의 길이에 따라서 질의 처리 성능이 달라질 수 있다.

또한 비트맵 필터는 Long List로 분류된 역 리스트들을 탐색할 때만 이용된다. 그리고 문자열 집합의 문자열들에 의해 발생되는 q-그램들의 발생 빈도는 Power-law를 따른다[3]. 따라서 발생 빈도가 높은 q-그램들의 역 리스트들이 Long List로 분류될 가능성이 크고 이와 같은 이유로 발생 빈도가 높은 q-그램의 역 리스트들에 대해서만 비트맵 필터를 구성하여도 질의 처리 성능의 저하 없이 비트맵 필터의 공간 비용을 줄일 수 있다.

Algorithm1 build bitmap filter
input : Inverted list I
output : Bitmap filter on inverted list I

```
01. Initialize a bitmap filter BFilter to 0's;
02. FOR(each ID in inverted list I)
03.     Group = ID / CSize * BLength;
04.     BFilter[Group] := BFilter[Group] bit-or 1;
06. RETURN BFilter;
```

(그림 3) 역 리스트에 대한 비트맵 필터 구성 알고리즘

3.3. 비트맵 필터를 이용한 역 리스트 탐색

(그림 2)와 같은 비트맵 필터를 구성하면 그룹 비트가 0일 경우에는 역 리스트에 문자열 ID가 포함될 가능성이 없으므로 그룹 비트를 공유하는 문자열 ID들은 역 리스트를 탐색하지 않아도 된다. 따라서 비트맵 필터를 이용하여 불필요한 역 리스트의 탐색을 회피할 수 있다.

비트맵 필터를 이용하여 문자열 ID가 역 리스트에 포함되어 있는지 확인하기 위해서 그 문자열 ID가 속한 그룹을 찾아야 하며 문자열 ID/GFactor번째 그룹에 포함된다. 문자열 ID가 속한 그룹을 찾은 후에 비트맵 필터에서 그룹의 비트를 확인한다. 만약 그룹 비트가 1인 경우에는 역 리스트에 문자열 ID가 포함될 가능성이 있으므로 역 리스트를 이진 탐색하여 해당 문자열 ID의 포함 여부를 확인한다. 그렇지 않은 경우에는 역 리스트에 문자열 ID가 포함되어 있지 않으므로 역 리스트를 탐색하지 않는다. (그림 4)는 비트맵 필터를 이용하는 역 리스트 탐색 기법을 적용한 리스트 병합 알고리즘을 기술한 것이다.

(그림 4)의 05번 줄에서는 Long List에 포함된 역 리스트들 중 문자열 ID의 그룹 비트가 1인 역 리스트들의 수를 확인한다. 이때 비트맵 필터를 가지지 못한 역 리스트들은 문자열 ID가 역 리스트에 포함될 가능성이 있으므로 그룹 비트가 1인 것으로 간주한다. 07번 줄에서는 비트맵 필터를 통해 확인된 문자열 ID가 포함될 가능성이 있는 역 리스트의 개수와 Short List에서 문자열 ID가 발생된 횟수를 이용하여 불필요한 역 리스트 탐색을 회피한다.

Algorithm2 list merging algorithm using bitmap filter
input : set of inverted list and a threshold T
output : ids that appear at least T times on lists

```
01. Initialize a result set R to be empty;
02. Let L_long be the set of T-1 longest lists among the lists;
03. Let L_short be the remainig short lists;
04. FOR(each record ID in merged L_short)
05.     Let L_count be the number of lists such that group bit of ID is 1;
06.     Let ID_count be the number of occurrence of r among L_short;
07.     IF(L_count + ID_count < T) continue;
08.     FOR(each list in L_long )
09.         Check if ID appear on this list using binarysearch;
10.         if ID appears T and more times among all lists,
              add ID to R;
11. RETURN R;
```

(그림 4) 비트맵 필터를 이용한 리스트 병합 알고리즘

4. 실험

이 장에서는 비트맵 필터를 사용하는 리스트 병합 알고리즘과 그렇지 않은 기존의 리스트 병합 알고리즘의 성능을 실험을 통해 비교한다. 실험은 Intel Dual-Core 3GHz, 2GB RAM, Ubuntu 10.04 LTS 32bit의 환경에서 수행하였고 3-그램을 사용하여 역 리스트들을 구성하였다. 유사도는 편집거리 2로 고정하였으며 1000개의 질의를 수행하여 측정하였다. 실험에 사용된 데이터는 <표 1>과 같다.

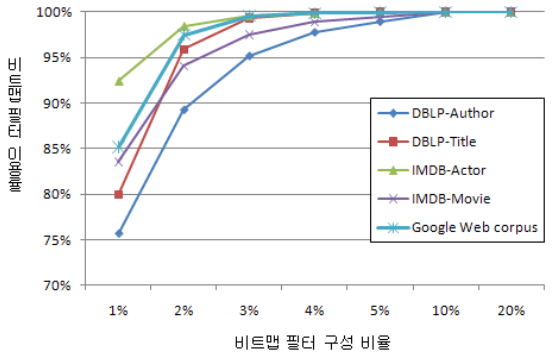
<표 1> 실험 데이터 집합

	문자열 개수	문자열 평균 길이
DBLP Author	2948930	15
DBLP Title	1158649	68
Google Web corpus	3000000	21
IMDB Movie	1568894	19
IMDB Actor	1213392	16

4.1. 비트맵 필터 구성 비율에 따른 비트맵 필터 이용률

비트맵 필터의 공간 비용을 줄이기 위해서 문자열 집합의 문자열들이 많이 포함하는 q-그램의 역 리스트에 대해서만 비트맵 필터를 구성하는 방법을 사용할 수 있다. (그림 5)는 비트맵 필터를 구성하는 비율을 다르게 하여 비트맵 필터 이용률 측정한 것이다. (그림 5)와 같이 전체 역 리스트 중 길이가 가장 긴 5%의 역 리스트들에 대해서 비트맵 필터를 구성하는 경우와 그 이상의 역 리스트에 대해서 비트맵 필터를 구성하는 경우는 비트맵 필터 이용률의 차이가 거의 없다. 따라서 전체 역 리스트들 중 가장 긴 5%의 역 리스트들에 대해서만 비트맵 필터를 구성해

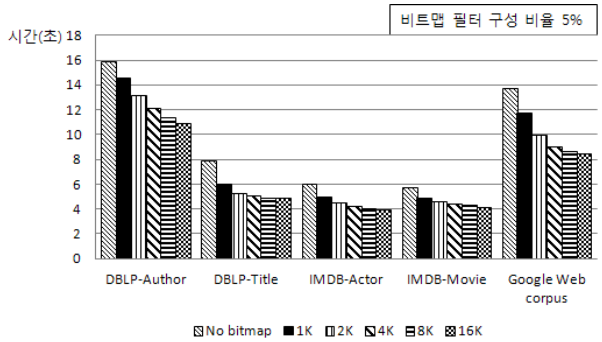
도 질의 처리 성능에는 영향이 없으므로 질의 처리 시간은 비트맵 필터 구성 비율을 5%로 하여 실험하였다.



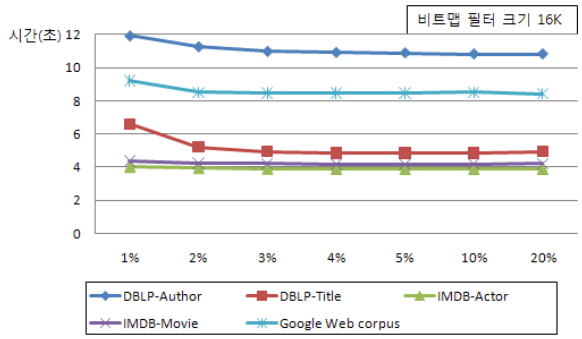
(그림 5) 비트맵 필터 구성 비율에 따른 비트맵 필터 이용률

4.2. 질의 처리 시간

(그림 6)은 비트맵 필터의 길이에 따른 질의 처리 시간을 보여주고 있다. 모든 데이터 집합이 필터 크기가 증가함에 따라 질의 처리 성능이 향상되었다. 16KB 비트맵 필터를 사용한 경우에는 비트맵 필터를 사용하지 않은 경우보다 약 30~40%정도의 성능 개선 효과가 있음을 보여주고 있다. 그리고 (그림 7)에서는 비트맵 구성 비율에 따른 질의 처리 시간을 보여주고 있다. 모든 데이터 집합이 비트맵 구성 비율이 5%이상이면 더 이상 질의 처리 시간의 변동이 없다는 것을 보여주고 있다.



(그림 6) 비트맵 필터 길이에 따른 질의 처리 시간



(그림 7) 비트맵 필터 구성 비율에 따른 질의처리 시간

5. 결론

본 논문에서는 효율적인 유사 문자열 검색을 위한 비트맵 필터를 이용한 역 리스트 탐색 기법을 제안하였다. 제안한 기법은 Long List로 분류된 역 리스트들을 탐색하기 이전에 비트맵 필터를 확인하여 불필요한 역 리스트 탐색을 회피하여 리스트 병합 비용을 절감하였다.

실험을 통하여 비트맵 구성 비율을 5%로 하였을 경우와 그 이상으로 하였을 경우 질의 처리 성능의 차이가 없음을 확인하였으며 비트맵 필터의 크기가 늘어남에 따라 질의 처리 시간이 감소함을 확인하였다. 그리고 제안된 기법을 적용한 리스트 병합 알고리즘이 30~40%정도의 성능 개선 효과가 있음을 실험을 통하여 확인하였다.

향후에는 비트맵 필터의 공간 비용을 더 줄이기 위해 역 리스트의 길이나 역 리스트내의 문자열 ID의 분포에 따라 비트맵 필터의 길이를 다르게 하는 역 리스트 탐색 방법에 대해 연구할 예정이다.

참고문헌

[1] S. Sarawagi and A. Kirpal, "Efficient set joins on similarity predicates", SIGMOD, pp743-755, 2004.

[2] Chen Li, Jiaheng Lu, and Yiming Lu, "Efficient Merging and Filtering Algorithms for Approximate String Searches", ICDE, pp.257-266, 2008.

[3] Chen Li, Bin Wang and Xiaochun Yang, "VGRAM: Improving Performance of Approximate Queries on String Collections Using Variable-Length Grams", VLDB, pp.303-314, 2007.

[4] Alexander Behm, Shengyue Ji, Chen Li and Jihang Lu, "Space-Constrained Gram-Based Indexing for Efficient Approximate String Search", ICDE, pp.604-615, 2009.

[5] Arvind Arasu, et al., "Efficient Exact Set-Similarity Joins." VLDB, pp.918-929, 2006.

[6] Marios Hadjieleftheriou, et al., "Hashed samples: selectivity estimators for set similarity selection queries." PVLDB, 2008.

[7] Chuan Xiao, Wei Wang, Xuemin Lin, "Ed-Join: an efficient algorithm for similarity joins with edit distance constraints", PVLDB, 2008.

[8] Xiaochun Yang, Bin Wang, and Chen Li, "Cost-Based Variable-Length-Gram Selection for String Collections to Support Approximate Queries Efficiently", SIGMOD, 2008.

[9] K. Chakrabarti, S. Chaudhuri, V. Ganti, D. Xin, "An Efficient Filter for Approximate Membership Checking", SIGMOD, 2008.

[10] Marios Hadjieleftheriou, Xiaohui Yu, Nick Koudas, Divesh Srivastava, "Hashed samples: selectivity estimators for set similarity selection queries." PVLDB, 2008.