평면 색인 구조에서 효율적인 k-근접 이웃 찾기

김 태완, 강 혜영, 이 기준
부산 대학교 전자 계산학과

Efficient k-nn search on directory-based index structure

Tae-Wan Kim, Hyeyeong-Kang, Ki-joune, Li
{twkim, hykang, lik}@quantos.cs.pusan.ac.kr

요약

최근에 제안된 VA-File[6]은 k-NN 질의 처리에서 아주 효율적이라고 알려져 있다. 제시된 방법은 분할된 데이터의 저장 효율성을 보장하지 못하기 때문에 각 차원에 할당된 비트의 수가 증가하면(비트수≈3~5) 할수록 거의 모든 데이터에 대하여 MBH를 생성하는 단점이 있다. k-NN 질의는 거의 모든 데이터를 순차 검색을 통한 일차적 가지제거작업을 한 후, 질의를 수행하기 위한 디스크 접근을 한다. 따라서, 질의를 수행하기 위한 디스크 접근 횟수는 다른 방법들에 비하여 거의 최적에 가까운 접근 횟수를 가지나 주 기억 장치에서 최소-힙을 이용하여 수행하는 일차적 가지 제거 작업의 오버 로더는 간과되었다. 우리는 기존에 알려진 재귀적으로 공간을 두개의 부 공간으로 분할하는 방법을 사용하여 VA-File 과 같은 디렉토리 자료구조를 구축하여 k-NN 실험을 하였다. 이러한 분할된 MBH의 정방형성을 선호하는 방법은 저장 효율성을 보장한다. 실제 데이터에 대한 실험에서 우리가 실험한 간단한 방법은 디스크 접근 시간 및 CPU 시간을 합한 전체 수행시간에서 VA-File에 비하여 최대 93% 정도의 성능 향상이 있다.

1장. 서론

최근 데이터 베이스와 관련된 많은 응용들은 고 차원이고 대용량 데이터에 대한 처리를 요구하고 있다. 새로운 응용 분야들은 멀티 미디어 시스템, 기상 및 환경 정보 시스템, 시뮬레이션 및 실험 결과 분석, 분자 생물학, 데이터 웨어하우징 이다. 이러한 응용 분야들에서 가장 중요하게 요구 되고 있는 기능들 중의 하나는 유사성 검색들 중 근접 또는 k-근접 이웃 질의이다. 근접 이웃질의에 대한 연구들은 이러한 질의처리를 효율적으로 처리 하기 위한 자료 구조를 제안한 방법들과 디스크에 기반을 둔 색인에서 질의를 처리 하는 방법들이 제안 되어 왔다. 본 연구에서 우리는 디스크에 기반을 둔 색인에서 근접 이웃 질의를 처리하는 방법들에 대하여 연구한다.

고 차원 데이터에 대한 색인 방법은 중간 또는 단말 노드를 구축하는 경우 모든 축에 대하여 분할을 수행할 수 없다. 예를 들면, 차원이 20 이고 생성 해야 하는 노드들의 수가 1000개인 경우, 각 차원에 대하여 한 번씩 분할 하더라도 $2^{20}(\approx 1000000)$개의 분할 결과들을 생성한다. 따라서, 1000개의 노드를 생성하기 위해서는 9~10개 차원에 대해서만 한번의 분할을 수행할 수 있다. 나머지 10~11개의 차원은 분할대상 축으로 선택되지 않는다. 생성하려는 노드의 개수가 줄어들면 들수록(루트 노드에 가까워 질수록) 분할 축의 수는 더욱 줄어들 것이다. 이러한 사실은 트리 형태를 가지는 색인의 중간 노드들은 질의에 대하여 분기력이 아주 저조함을 알 수 있다. 높은 차원에 대해서는 단말 노드들의 질의에 대한 분기력 역시 저조할 것이다. 따라서, 다 차원 데이터에 대하여 계층을 가지는 색인보다는 평면적인 색인 구조의 성능이 더 좋을 수 있으며 이러한 사실은 [6]에서 보여 주었다. 평면 색인 구조의 장점은 요약정보의 크기는 주 기억장치에 적재할 수 있을 정도로 작다는 것이다. 따라서, 질의 처리는 적재된 요약정보에 대한 여과 작업을 한 후, 실제 데이터에 대한 디스크 접근을 한다. VA-File에서 HS-알고리즘[3]을 평면 색인 구조에 적용이 가능하도록 변형 및 개선한 NOA-알고리즘[6]을 제안하였다. NOA-알고리즘은 디렉토리의 엔트리를 질의 점에 대한 최소 거리로 최소-힙을 구축한 후, 최소 및 최대 거리를 이용하여 k 개의 근접 이웃들을 찾는다. 이 방법은 엔트리들을 질의 점과 먼 순서부터 최소 힙에 삽입 하는 경우 모든 엔트리들을 최소-힙에 저장하는 단점이 있다. 그리고 VA-File은 저장 효율성을 보장하지 못하기 때문에 거의 모든 데이터에 대하여 하나의 엔트리를 생성할 수도 있다. 따라서, VA-File은 최악의 경우 모든 데이터에 대하여 우선적으로 거리에 따른 정렬을 수행하는 단점이 있다. 이러한 단점들은 과도한 CPU 시간 비용을 초래한다. 우리는 본 연구에서 이러한 단점을 극복하는 세 단계 근접 이웃 알고리즘인 TP(Three Phased)-알고리즘을 제안한다. 그리고, 제안하는 알고리즘의 정확성에 대해서도 보여준다. NOA-알고리즘에 대한 단순한 변화인 TP-알고리즘을

적용한 실험은 아주 상당한 성능의 향상을 보여준다. 균등-개수 또는 STR[4] 방법을 이용하여 구축한 평면 색인의 실험에서 TP-알고리즘은 NOA에 비하여 최대 93% 정도의 수행시간만을 필요로 한다.

본 논문은 다음 장에서 평면 색인 구조의 일반적인 특징에 대하여 기술하고 본 연구에서 사용된 간단한 평면 디렉토리 생성에 대하여 기술한다. 그리고, 두 개의 잘 알려진 근접 이웃 알고리즘인 RKV 알고리즘[5]과 HS 알고리즘을 간단히 소개 한다. 그리고, 3장에서 VA-File 에서 제시한 NOA 방법 및 우리의 TP 방법에 대하여 기술한다. 그리고 TP의 정확성에 대해서도 언급한다. 4장에서는 평면 색인 구조에 대한 근접 이웃 질의를 수행한 실험 결과를 보여주고 5장에서 본 연구를 결론 짓는다.

2장. 평면 색인 구조들

평면 색인 구조는 데이터를 두 단계로 나누어서 저장한다. 분할된 데이터는 디스크 페이지에 저장이 되며 디스크에 저장이 된 데이터에 대한 요약 정보(MBH, 최소 경계 하이퍼 입방체)는 주기억 장치에 저장된다. 이 때, 요약정보의 집합을 디렉토리 라고 한다. 우리는 비교의 편의를 위하여 디렉토리의 엔트리는 (MBH, 디스크 페이지주소, 페이지개수) 구조를 가진다고 가정한다. 즉, MBH에 속한 모든 데이터는 디스크 페이지에 저장된다고 가정한다. 평면 색인 구조의 한 예는 VA-File이 될 수 있다. VA-File 은 우선 각 차원에 대하여 분할 개수 2^b를 결정한다. 따라서, d 차원의 경우 2^{bd}개로 분할된 그리드 셀들이 가상적으로 존재한다고 가정하고 각 셀에 데이터가 포함되면 셀에 대한 요약 정보를 생성하여 디렉토리에 삽입한다. 할당된 비트의 크기 b는 3~5 정도인 경우 근접질의에서 좋은 성능을 보여준다고 [6]에서 지적하였다. 그리고, 각 차원에서의 2^b 개의 분할선의 선택 방법에 대한 자세한 기술은 없으므로 우리는 각 차원에 대하여 동일한 개수를 가지는 2^b -1개 분할선을 이용하였다. 색인에 근거한 k-NN 알고리즘들은 RKV-알고리즘, HS-알고리즘들이 있다. RKV-알고리즘에 대한 향상된 방법은 [2]에서 제안 되었으며 HS-알고리즘의 최적의 디스크 접근 회수를 가진다고 [1]에서 증명 되었다. RKV-알고리즘과 HS-알고리즘은 트리 형태의 색인 구조를 가지는 자료구조에 대하여 적용이 되었다. 평면 색인 구조인 경우 두 방법을 적용한 경우, RKV-알고리즘은 디렉토리에 있는 엔트리들을 접근하는 순서대로 디스크에 있는 데이터를 접근한다. HS-알고리즘은 디렉토리에 있는 엔트리들을 최소-힙 자료구조를 이용하여 질의 벡터 q 에 대하여 거리가 짧은 순으로 정렬한 후, 디스크에 있는 데이터를 접근한다.

균등-개수 분할은 어떤 차원에서 데이터를 동일한 개수로 이진 분할하여 분할된 데이터의 개수가 정해진 개수가 될 때 까지 재귀적으로 반복한다. 가장 긴 길이를 가지는 차원을 분할 차원으로 선택한다. 이러한 방법을 *EC-방법*이라 한다.

3장. NOA 및 TP-알고리즘

NOA는 두 단계로 구성되어 있는데 첫번째 단계에서 엔트리들과 질의 점 사이의 최소거리를 구한 후 이를 최소-힙에 삽입한다. 그리고, 최소-힙에 삽입하는 엔트리는 선택된 k 개의 엔트리들의 최대 거리보다 최소거리가 짧은 경우에만 힙에 삽입된다. 따라서, 최소-힙은 질의 점과 최소거리가 짧은 순으로 엔트리들을 정렬한다. 두 번째 단계에서는 최소-힙에 있는 순서대로 엔트리들을 하나씩 처리한다. 이때, 선택된 k 개의 데이터보다 엔트리의 거리(질의 점과의 최소거리)가 크면 NOA은 k 개의 최 근접 이웃을 가진다. TP 및 NOA는 공통적으로 표 1에 있는 함수를 이용하여 기술되며 NOA 방법은 [6]에 상세히 기술되어 있으며 TP는 그림 1에 나타내었다.

표 1. 공통된 함수들

함수	설명
L2_Dist(q, x)	점 x 와 q 사이의 L_2 거리 값
InitCandidate(L)	리스트 L의 초기화. L의 크기는 k 이며 FLT_MAX 값을 반환한다.
Candidate(L, e)	리스트 L의 최대값보다 e의 거리 값이 작으며 e를 L에 삽입하고 L의 최대 값을 반환한다.
Resolve (L,q,p_addr)	p_addr에 있는 데이터들과 q와의 L2_Dist를 이용하여 Candidate함수를 수행한다. (디스크접근함수)

```
TP-Algorithm
단계 1-1:
page_addr = q를 포함하는 entry의 주소;
p = LoadPage(page_addr);
For all e_j ∈ p
    elem.dist = L2_Dist(q, e_j); elem.mbh = e_j;
    δ = Candidate(CAND_LIST, elem);
단계 1-2:
for all entries {
  if(entry[i].page_addr != page_addr){
    elem.dist = MinDist(q, entry[i].mbh);
    if(elem.dist < δ){
      elem.page_addr = entry[i].page_addr;
      InsertHeap(Heap, elem);
}}}
단계 2:
elem = PopHeap(Heap);
While (elem.dist < δ){
  δ = Resolve(CAND_LIST,q, elem.page_addr);
  elem = PopHeap(Heap);
}
```

그림 1. TP 알고리즘

정리. TP 방법을 k-개의 근접 이웃을 찾는다.

증명. 단계 1-2에서 엔트리와 q 와의 MinDist가 CAND_LIST의 최대값보다 작은 경우에는 CAND_LIST 에 있는 k 개의 데이터보다 더 거리가 짧은 데이터가 존재할 수 도 있다. 따라서, 이러한 엔트리는 힙에 삽입된다. 그렇지 않은 경우, 우리는 이미 k 개의 데이터를 찾았기 때문에 힙에 삽입할 필요가 없다. □

4장 실험.

우리는 개수(N=68040)가 동일한 실제 데이터 집합에 대한 실험을 하였다. 9-CM은 9 차원의 색상 모멘트 데이터이고 16-CT는 16차원의 색상 질감 데이터 이고 32-CH 데이터는 32차원의 색상 히스토그램 데이터이다. 이러한 데이터 집합들은 여러 연구들에서 실험에 공통적으로 많이 이용한다. 질의 집합은 각각 데이터 집합에서 임의로 1000개의 점 데이터를 추출한 후, 이들을 이용하여 k 근접 질의를 수행하였다. 개수 k의 값은 10으로 하였다. NOA는 VA-File에 의하여 생성된 엔트리에 대하여 적용 되었고, TP는 EC 방법을 적용하였다. 결과는 표2에서 보여준다. EC 방법에 NOA를 적용한 실험 결과 힙의 크기는 TP 방법에 비하여 약 10배 정도 더 많다. 따라서 증가된 부분 만큼의 CPU 시간 비용을 더 초래한다.

표 2.실험 결과(|E|,|H|:엔트리,힙의 크기,U:저장효율성)

차원	방법	1단계 (|E|)	2단계 (|H|)	실행시간 (U)	NOA-TP / NOA
9	NOA (b=2)	171.5 (10211)	34.2 (7399.9)	205.7 (2.9%)	0.49
	TP	**46.4** (300)	**58.2** (26.6)	**104.6** (100%)	
16	NOA (b=2)	370.8 (15986)	136.9 (9886.3)	507.7 (3.4%)	0.75
	TP	**58.0** (536)	**71.3** (133.5)	**129.3** (100%)	
32	NOA (b=1)	*2758.5* (48522)	*2433.5* (48522)	*5192.0* (2.3%)	0.93
	TP	**79.31** (1080)	**289.86** (175.42)	**365.27** (100%)	

할당된 비트의 수가 증가하면 실행시간은 줄어드나 저장 효율성을 떨어진다. 비트수가 4(=2^4-1개의 분할선)인 경우의 TP와의 비교 실험에서도 TP방법은 여전히 25%, 49%, 48%의 성능향상이 있다.

5장 결론.

디렉토리 구조를 가지는 색인에서 근접 이웃질의를 처리하는 보다 빨리 처리 하는 방법에 대하여 본 연구는 기술하였다. 알고리즘의 초기 단계에서 하나의 엔트리에 포함되는 데이터를 처리하여 구성한 하이퍼 구에 의하여 질의 처리를 위하여 사용되는 힙 자료 구조에 삽입되는 엔트리들의 수를 최소화 하였다. 이러한 CPU 비용의 절감은 디스크 접근 비용을 충분히 보상한다.

참고문헌

[1]. C. Bohm, S. Berchtold, D.A. Keim, "*Searching in High-Dimensional Spaces - Index Structures for Improving the Performance of Multimedia Databases*", ACM Computing Surveys, 33.3. pp. 322-373, 2001.

[2]. K. Cheung and A. Fu, "*Enhanced nearest neighbor search on the r-tree*", SIGMOD Rec. 27.3.

[3] G.R. Hjaltason and H. Samet, "Ranking in Spatial Databases", SSD, 1995, pages 83-95

[4]. S.T. Leutenegger, M.A. Lopez and J. Edington, "*STR: A Simple and Efficient Algorithms for R-Tree Packing*", ICDE, pp 164-171, 1997.

[5]. N. Roussopoulos, S. Kelly, and F. Vincent, "*Nearest Neighbor Queries*", In Proc. Of ACM SIGMOD Conference, 1995, pages 71-79

[6] R. Weber and S. Blott, "*An Approximation-Based Data Structure for Similarity Search*", TR-24, ESPRIT project HERMES #9141, 1997.