대용량 멀티미디어 데이터베이스를 위한 LSH 파라메터 실험
A Study on LSH Parameters for Large Multimedia Databases

홍 지 원, 문 병 문, 김 상 욱
한양대학교

Jiwon Hong, Byung-Moon Moon, Sang-Wook Kim
Hanyang University

요약

LSH는 고차원 데이터베이스에서의 빠른 유사 아이템 검색을 위해 널리 사용되고 있는 인덱싱 방안이나, 다양한 파라메터가 존재하여 각 파라메터를 적절하게 설정하는 데에 어려움이 있다. 본 논문에서는 다양한 실험을 통해 고차원의 대용량 멀티미디어 데이터베이스에서의 유사 아이템 검색을 위한 LSH의 파라메터에 따른 성능 추이를 살펴보고, 적절한 파라메터를 설정하는 방안에 대해 논의한다.

Ⅰ. 서론

오늘날 우리는 멀티미디어 데이터의 홍수 속에 살고 있다. 멀티미디어 데이터에는 비디오, 오디오, 이미지 등이 있으며, 대부분 고차원의 벡터로 표현될 수 있다. 이러한 멀티미디어 데이터를 저장하는 데이터베이스는 갈수록 대용량화되어가고 있는 추세이며, 이에 따라 대용량고차원 멀티미디어 데이터베이스에서의 빠른 인덱싱은 중요한 이슈로 연구되고 있다[1].

대용량고차원 멀티미디어 데이터베이스에서의 유사 아이템 검색에는 기존에 널리 사용되던 R* 트리, SR 트리 등의 트리 기반 인덱싱 방안이 사용되기 어렵다. 기존 트리 기반 인덱싱 방안은 k-최근접 유사 아이템 검색에서 정확한 결과를 제공할 수 있으나, 차원의 수가 높아질 경우 검색 속도가 심각하게 감소하는 차원의 저주(curse of dimensionality)에 직면하게 된다.

이러한 문제를 피할 수 있는 방안으로 LSH(Locality Sensitive Hashing)를 이용한 인덱싱 방안이 제안되었다[2]. LSH는 해쉬(hash)의 특징을 이용하는 유사 아이템 검색 방안으로, 트리 기반 인덱싱 방안이 정확한 k-최근접 유사 아이템 검색 결과를 제공하는 데 반해 근사 k-최근접 유사 아이템 검색 결과를 제공한다. LSH는 해쉬를 이용한 일반적인 인덱싱 방법과 달리 정확하게 동일한 값만이 같은 해쉬 값을 갖도록 하는 대신 유사한 값들이 같은 해쉬 값을 갖도록 하는 해쉬 함수를 이용한다. 고차원 데이터의 아이템들 중 같은 해쉬 값을 갖는 유사한 아이템의 범위를 정하는 것은 쉽지 않은 일이다. LSH는 이러한 어려움을 임의로 생성된 여러 개의 해쉬 함수와 해쉬 테이블을 이용하여 해결한다. 이를 위해 [2]에서 제안된 LSH는 해쉬 함수 및 해쉬 테이블의 수, 해쉬 버킷의 수, 해쉬 버킷의 크기를 주요 파라메터로 갖는다. 또한 해쉬 함수 집합을 생성하는 데에 사용되는 변수 k 역시 주요 파라메터의 하나이다. LSH는 기존 트리 기반 인덱싱에 비해 그 유용성이 증명되어 있으나 [1][2][4], 주요 파라메터의 변화에 따라 근사 k-최근접 유사 아이템 검색의 정확도 및 검색 속도에 큰 차이가 나기 때문에, 주요 파라메터를 적절하게 선택하는 데에 어려움이 따른다.

본 논문에서는 LSH의 주요 파라메터들의 변화에 따른 정확도 및 검색 속도 변화의 추이를 이미지 데이터를 이용한 실험을 통해 알아보고 이로부터 주요 파라메터를 설정하는 데에 필요한 통찰을 얻고자 한다.

Ⅱ. Locality Sensitive Hashing

LSH는 해쉬의 특징을 이용하기 때문에 이에 사용되는 해쉬 함수가 인덱싱 성능의 상당 부분을 결정한다. LSH에서는 대표적으로 해밍 거리(Hamming distance)를 이용하는 해쉬 함수 집합[2]과 p-Stable 분포를 이용하는 해쉬 함수 집합[4]이 주로 사용되고 있다.

해밍 거리를 이용하는 LSH는 주어진 데이터를 해밍 공간(Hamming space)에 매핑하여 실제 데이터를 다음과 같은 이진 표현으로 나타낸다.

$$v(p) = \mathrm{Unary}_C(x_1) \ldots \mathrm{Unary}_C(x_d) \qquad \text{(수식 1)}$$

이 때, 실제 데이터 p는 d개의 값을 갖는 벡터이며, 각 차원의 값은 모든 데이터 값 중 가장 큰 값 C 길이의 2진수로 변환된다. p의 이진 표현 $v(p)$는 이러한 C 길이의 2진수 d개를 결합한 길이 dC의 2진수로 표현된다. 각 차원의 값은 해당 값만큼의 연속된 1과 C에서 해당 값을 뺀 만큼의 연속된 0으로 이루어진다. 해밍 거리를 이용하는 해쉬 함수 집함은 이렇게 변환된 p의 이진 표현으로부터 미리 임의로 정해진 k개의 비트를 선택하는 방식으로 정의된다.

해밍 거리를 이용하는 LSH는 이와 같이 해밍 거리 해쉬 함수를 L개 정의하고, 각각의 해쉬 함수에 상응하는 해쉬 테이블을 L개 생성한다. 실제로 어떤 데이터 아이템 p가 삽입되었을 때, LSH는 L개의 해쉬 테이블에 각각 p를 삽입한다. 이렇게 전체 데이터에 대해 L개의 해쉬 테이블을 이용한 인덱스를 생성한 이후 쿼리 데이터 아

이템 q가 주어지면 q를 L개의 해쉬 함수를 통해 해쉬하여 나온 해쉬 값에 해당하는 L개의 해쉬 버킷들에 들어있는 아이템들을 후보 아이템들로 간주한다. 이러한 후보 아이템들의 수는 전체 아이템 수에 비해 매우 적기 때문에, 각 후보 아이템들과 q 간의 거리를 계산하여 실질적인 k-최근접 아이템을 구한다.

p-Stable 함수를 이용하는 LSH는 해쉬 함수를 구성하기 위한 k개의 요소를 각각 다음 식을 통해 계산한다.

$$h_{a,b}(v) = \lfloor \frac{a \cdot v + b}{r} \rfloor \qquad \text{(수식 2)}$$

이 때, 주어진 데이터의 벡터 표현 v를 미리 임의로 정해진 같은 길이의 벡터 a와 역시 미리 임의로 정해진 상수 b를 이용하여 하나의 값으로 바꾸고, 이를 r개의 구간으로 나누어 해쉬한다. p-Stable 함수를 이용하는 LSH에서는 이러한 함수에 의한 해쉬 값 k개를 합친 것이 하나의 LSH 해쉬 함수에 해당한다.

Ⅲ. 실험 및 결과

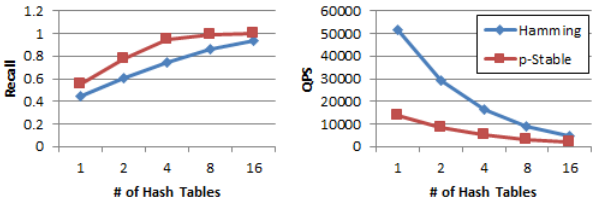
본 연구에서는 LSH를 이용한 고차원 멀티미디어 데이터베이스 인덱싱의 파라메터 변화에 따른 검색 정확도 및 속도를 알아보기에 앞서 우선 해밍 거리를 이용한 LSH와 p-Stable 함수를 이용한 LSH 간의 성능을 비교해 보았다. 이를 위해 우선 [3]에서 사용된 것과 동일하게 임의로 생성된(synthetic) 데이터에 대해 k와 L 값을 변화시키며 두 LSH를 비교하였다 (그림 1-2) 전체적으로 p-Stable 함수를 이용한 LSH가 약간 더 높은 정확도를 보였으나 해밍 거리를 이용한 LSH가 큰 차이로 더 빠른 검색 속도를 보였다. 따라서 본 연구에서는 해밍 거리를 이용한 LSH를 통해 고차원 멀티미디어 데이터베이스를 인덱싱하는 것이 유리하다고 판단하였다.

실제 고차원 멀티미디어 데이터베이스의 상황에서 실험을 수행하기 위해 본 연구에서는 SURF descriptor [5]를 이용하여 약 230만 개의 SURF descriptor를 갖는 이미지 데이터를 생성하였다. 실제 데이터를 이용한 실험에서는 해밍 거리를 이용한 LSH만을 사용하였다.

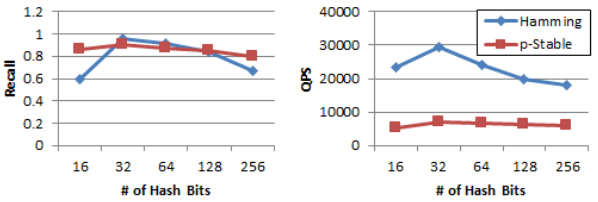
그림 3-4는 각각 LSH의 주요 파라메터인 k, L 값을 변화시키며 검색 정확도 및 속도를 recall과 QPS로 알아본 결과이다. 그림 3에서, k가 변화함에 따라 정확도가 크게 상승하나, 일정 수준을 넘어가게 되면 같은 버킷에 들어가게 되는 후보 아이템의 수가 크게 감소하게 되어 정확도가 오히려 떨어지는 것을 확인할 수 있었다. 또한 속도는 k의 증가에 따라 지속적으로 감소하였다. 그림 4에서도 역시 k가 일정 수준 이상일 경우 L에 증가함에 따라 오히려 정확도가 감소하는 경향을 발견하였으며, 속도도 지속적으로 감소하였다. M과 B는 정확도에는 큰 영향을 미치지 않았으나 속도와 밀접한 관계가 있음을 확인할 수 있었다.

Ⅳ. 결론

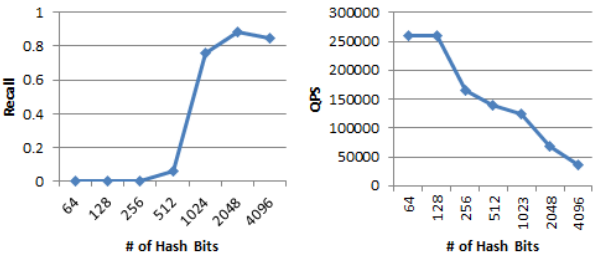
본 연구에서는 실험을 통해 해밍 거리를 이용한 LSH와 p-Stable 분포를 이용한 LSH를 비교해 보았으며, 더 빠른 검색 속도를 보인 해밍 거리 LSH에 대해 다양한 실험을 통해 각 파라메터에 의한 검색 정확도 및 검색 속도의 변화 추이를 살펴보았다.



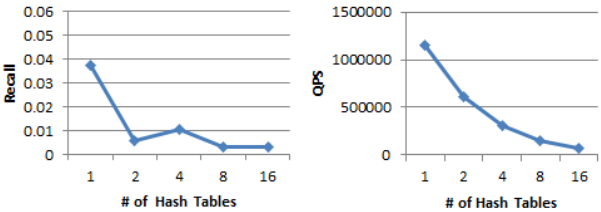
▶▶ 그림 1. k에 따른 정확도와 속도 (해밍 vs. p-Stable)



▶▶ 그림 2. L에 따른 정확도와 속도 (해밍 vs. p-Stable)



▶▶ 그림 3. k에 따른 정확도와 속도 (SURF 데이터)



▶▶ 그림 4. L에 따른 정확도와 속도 (SURF 데이터)

■ 감사의 글 ■

본 논문은 서울시 지원으로 수행한 「서울시 창조전문인력 양성사업 (HM120006)」의 결과입니다.

■ 참 고 문 헌 ■

[1] Bertino, E., et al. Indexing techniques for advanced database systems. Springer Publishing Company, Incorporated, 2012.

[2] Indyk, P. and Motwani, R., "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," Proc. of the ACM Symp. on Theory of Computing, pp. 604-613, 1998.

[3] 문병문 외, "고차원 멀티미디어 데이터 인덱싱을 위한 Locality Sensitive Hashing", 한국정보과학회 2013 추계학술발표회 눈문집, pp. 302-303, 2013.

[4] Datar, M., Immorlica, N., Indyk, P. and Vahab, S. M., "Locality-Sensitive Hashing Scheme Based on p-Stable Distributions," Proc. of the Symp. on Computational Geometry, pp. 253-262, 2004.

[5] Bay, H., Tuytelaars, T. and Gool, L. V., "SURF: Speeded UP Robust Features," Proc. of the European Conf. on Computer Vision, pp. 404-417, 2006.