사용자 선호도 추론 기반 인터랙티브 Top-K 질의 처리 기법

Interactive Top-K Query Based on User Preference

요 약

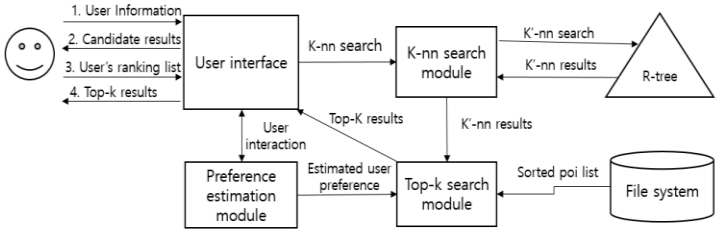
본 논문은 사용자의 위치와 선호도를 기반으로 주변 관심지점(Points Of Interests)를 검색하는 시스템을 설명한다. 사용자의 위치에서 가까운 관심지점을 찾는 기존의 질의 처리 시스템은 사용자의 개별 선호도를 반영할 수 없다는 문제가 있는 반면, 본 시스템은 별도의 기준을 추가한 가상의 데이터로 선호도를 추론하고, 이를 반영한 Top-K 최 근접 질의 처리를 진행한다. 따라서, R-tree기반의 최 근접 질의처리와 Threshold Algorithm을 이용해 시스템 환경을 구현하였다. 본 논문을 바탕으로 개발된 시스템은 사용자마다 선호도가 다름을 인지하고, 사용자와 상호작용하는 인터페이스에서 샘플데이터를 통해 개별 사용자의 선호도를 추론하고, 보다 개인화된 결과를 제공할 수 있다.

1. 서 론

최근 모바일 디바이스의 보급과 함께 사용자의 위치를 기준으로 가장 가까운 관심지점(Points of Interests, POI)를 찾아주는 위치 기반 검색 시스템들이 상용화되었다. 이와 더불어 사용자에게 만족도 높은 서비스를 제공하기 위해 위치 키워드 질의, Top-k 질의 등 다양한 속성들을 결합한 많은 연구가 진행되고 있다[1,2,3]. 하지만 기존 연구의 경우 고정된 선호도를 적용해 모든 사용자에게 동일한 결과를 제공한다는 단점이 있다. 실제 서비스에서는 사용자마다 개별 속성(위치, 평점, 방문 횟수 등)에 대한 선호하는 정도가 다르고, 자신의 선호도를 직접 수치화할 수 있는 사용자는 많지 않다. 따라서, 본 논문에서는 사용자와의 상호작용을 통해 선호도를 추론하여, 보다 개인화된 결과를 얻을 수 있는 기법을 제안한다. 또한, 실제 POI를 후보로 선정하는 기존 기법과 달리, 가상의 POI를 생성해 샘플 데이터로 제공함으로써 처리시간을 단축하였다. 제안하는 시스템의 성능을 평가하기 위해 본 논문에서는 특정 지역 POI추천과 리뷰 공유 지역 기반 소셜 네트워크 시스템으로, 위치 기반 관련 연구에서 주로 사용되는 "Yelp"의 오픈 데이터 셋[i]을 사용했다. 제안 기법은 사용자의 위치정보를 바탕으로 R-tree를 이용한 최 근접 질의 처리를 한 후, 정렬된 POI 리스트와 함께 Threshold Algorithm(TA)을 이용해 Top-K 탐색을 진행한다[1,2,3,4]. 또한, 사용자와의 상호작용을 통해 선호도를 추론함으로써 사용자의 만족도를 높일 수 있는 기법을 제안하였다.

2. 관련 연구

관련 연구[3]에서는 Top-k질의 성능을 향상시키기 위해 연속된 질의를 처리하는 과정에서 캐시 형태로 저장된 기존의 질의 결과들을 사용함으로써 처리 시간을 단축한 연구를 진행했다. 그러나 본 연구에서는 각 사용자마다 한번의 질의 처리를 가정하였으므로 일반적인 TA을 이용하였다. [1]에서는 사용자와의 상호작용을 통해 선호도를 추정할 때, Sky-band 개념을 사용해 실제 POI 데이터 중에서 후보를 선정하는 방식을 적용했다. 하지만 이를 선정하는 과정에서 시간이 많이 걸리고, 실제 POI 분포가 고르지 않을 수 있다는 단점이 있다. 따라서 본 논문에서는 사용자 선호도 추론 시 기존의 POI가 아닌 가상의 샘플 데이터를 생성하여 최초 질의를 통해 사용자에게 제공하고, 그 순위를 입력 받아 선호도를 추론하는 방법을 제안한다.



[그림1] 시스템 구성도

3. 사용자 선호도 추론 기반 Top-K 질의처리 기법

본 시스템은 데이터 전처리 및 모듈 연결, 사용자와의 상호작용을 하는 User Interface Module, 사용자의 위치 정보를 기반으로 최 근접 질의 처리를 하는 K-Nearest Neighbor Module, 사용자가 입력한 샘플데이터 순위를 통해 선호도를 추정하는 Preference Estimation Module, 추론된 선호도와 최 근접 질의 처리를 결과를 바탕으로 우선순위 큐를 활용한 TA을 통해 최종 Top-K 결과를 찾는 Top-K Search Module로 구성되었다. 전체적인 시스템 구조는 [그림1]과 같다. 최 근접 질의 처리에서는 Top-K 탐색 모듈에서 반환하는 결과(k)보다 더 많은 개수(k')의 질의 처리를 진행하도록 설계했다. 그 이유는 Top-K 탐색 모듈에서 찾은 최종 결과의 개수가 k개보다 적을 경우, 다시 최 근접 질의 처리를 진행하여야 하는 상황이 발생할 수 있다. 이 때, 최 근접 리스트의 마지막 객체를 이용해 Top-K를 탐색할 경우, TA의 탐색 단계가 늘어나게 된다. 간단한 예시를 들어 [그림2]의 top-k list는 [(F3, 15.1), (F2, 15.1), (F1, 13)]으로 임계 값은 13보다 작거나 같아야 한다. [그림 2]에서 case1의 경우 top-3 탐색을 하기 위해서는 10단계를 거쳐서 F3의 A score와 F1의 B score로 top-k를 끝낼 임계 값을 구한다. 하지만 case2의 경우 F4의 존재로 인해서 4단계만에 임계 값을 구할 수 있다. 따라서 최 근접 질의처리를 통해 얻는 객체의 개수를 k보다 큰 k'로 설정하였다. 또한, 본 논문에서는 사용자 선호도 추론을 위한 점수 함수를 다음과 같이 정의하였다.

$$\text{score} = \alpha * sd + (1-\alpha) * sn \quad \text{.....[수식1]}$$

점수 함수에서 sd는 정규화된 거리 점수(distance score)를 의미한다. sn은 수치화할 수 있는 속성 데이터(numeric data score)를 정규화한 점수이다. α는 사용자의 선호도(sn에 비해 sd를 선호하는 정도)를 의미한다. 이 점수 함수를 계산하는 과정을 '점수화'라고 정의한다.

3-1. 데이터 전처리 및 최 근접 질의

데이터 전 처리 과정에서는 위치 정보(위도와 경도)를 2차원의 R-tree 인덱스에 저장한다. 또, 수치화할 수 있는 속성을 기준으로 POI들을 내림차순으로 정렬하여 저장한다. 본 논문에서는 이 속성을 사용자들이 실제 사용하는 평가 요소인 리뷰 개수로 선정하였다. 인터페이스에서는 사용자로부터 위치 정보와 Top-k의 k값(반환할 결과의 개수)을 입력 받는다. 필요한 모듈들을 순서에 맞게 호출하여 실행한 후, 최종적으로 탐색이 완료된 결과를 사용자에게 제공하면서 본 시스템은 종료된다. 최 근접 질의는 질의 위치로부터 가장 가까운 이웃 객체를 탐색하는 것으로 3절(시스템 구성)에서 언급했듯이 k'-최 근접 질의를 통해 얻은 리스트를 거리를 기준으로 내림차순으로 정렬하였다.

3-2. 선호도 추론

제안 시스템에서는 서론에서 언급했듯이 사용자의 선호도를

Score= A+B

A list case1

tid	A
F1	10
F2	9.1
F3	9

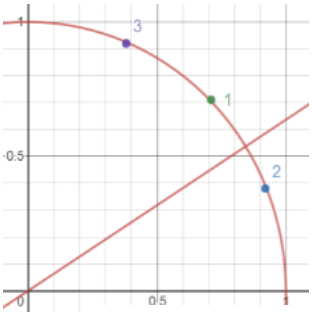
A list case2

tid	A
F1	10
F2	9.1
F3	9
F4	2

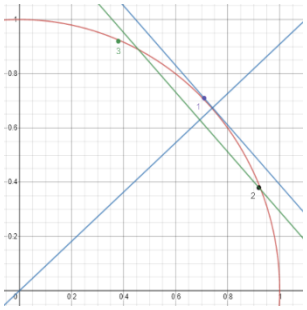
B list

tid	B	A
F10	9	1
F9	8	1
F8	7.4	1
F7	7.3	1
F6	7.2	1
F5	7.1	1
F4	7	2
F3	6.1	9
F2	6	9.1
F1	3	10

[그림2] Top-3 search 예시 case1과 case2는 B list는 같지만 A list의 길이는 다르다.



[그림 3-a]



[그림 3-b]

추론하기 위해 샘플데이터를 생성할 때, [그림 3]의 그래프($x^2 + y^2 = 1(0 \le x \le 1, 0 \le y \le 1)$)위의 점들로 구성했다. 원 위의 점들은 모두 원점으로부터 같은 거리에 위치하고 있으며, 임의의 두 점을 이은 직선 위에 다른 어떠한 점도 위치하지 않는다. 샘플데이터 개수를 나타내는 변수 n을 정의하여, 원의 둘레에 n개의 좌표를 생성한다. 이렇게 생성된 좌표들을 2개씩 짝지어 순서쌍을 만들고, 이 두 점을 지나는 직선의 수직인 벡터를 구한다. 이 벡터가 원점(0,0)을 지날 때, 임시 점수 함수라 정의하고, 이는 점수화 했을 때, 두 점이 동일한 점수를 가지게 되는 함수이다. 예를 들면, [그림 3]는 사용자가 정한 순위를 각 점에 대입한 모습이다. [그림 3-a]는 점 1과 2의 임시 점수 함수를 구한 모습이다. 이 임시 점수 함수에 대하여 점수화하면 다음과 같다.

$$\text{점 1의 점수} = \left(\frac{7}{11}\right)(0.92) + \left(\frac{4}{11}\right)(0.38) = 0.72$$

$$\text{점 2의 점수} = \left(\frac{7}{11}\right)(0.71) + \left(\frac{4}{11}\right)(0.71) = 0.71$$

[그림 3-b]는 임시 점수 함수의 벡터를 순위가 높은 점 쪽으로 기울인 모습이다. 두 점을 새 임시 점수 함수에 대하여 점수화하였을 때, 점 1이 더 큰 점수를 가지게 된다. (계산식 생략.) 이와 같이 모든 샘플데이터 순서쌍에 대하여 순위가 높은 쪽의 벡터의 범위(이상 또는 이하)를 정한다. 이상의 범위를 갖는 벡터 중 최댓값, 이하의 범위를 갖는 벡터 중 최소값을 선택해 이들의 중앙값을 구한다. 결론적으로 도출한 중앙값(α')을 사용자의 최종 점수 함수의 벡터로 설정한다. 따라서, [수식 1]에서 α(선호도)는 다음과 같다.

$$\alpha = \frac{1}{(\alpha' + 1)} \quad \text{.....[수식 2]}$$

3.3 Top-K 질의 처리

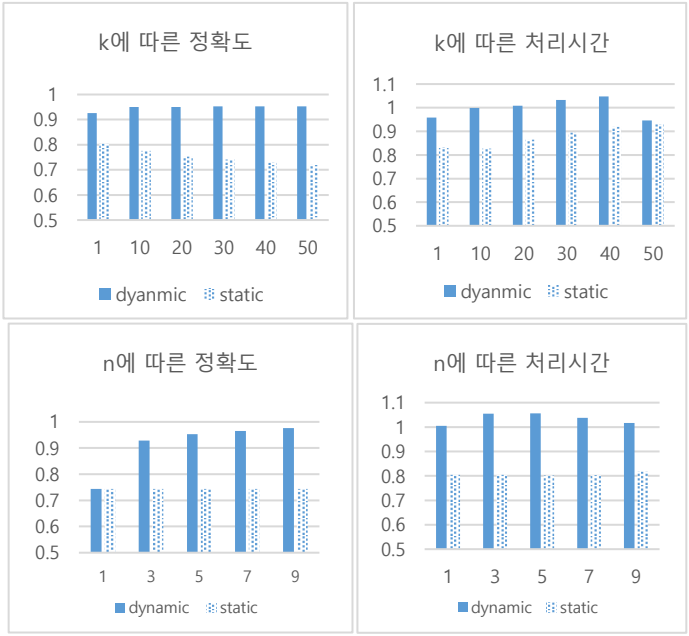
Top-k 질의처리에서는 각 객체들의 거리 값과 수치화할 수 있는 속성값을 정규화 한 후 점수 함수에 대입하여 점수화 한다. 그 다음 점수들을 기준으로 순위를 구하는데, 본 논문에서는 이를 위해 TA을 사용하였다. 해당 알고리즘에서는 임계 값을 이용해서 각 객체들의 점수와 비교하여 Top-K 리스트를 만들어낸다. 본 논문에서는 최 근접 질의 리스트와 정렬된 POI 리스트(3.1절)를 사용하여 Top-K 탐색을 진행한다. 자세한 과정은 다음 의사코드와 같다.

Top-K(k, knn list, POI list)

K = knn list;
P = POI list sorted by numeric data;
top-K Buffer = Priority Queue;
for i **to** size of P **do**
  if i > k' **then** K-item = k'-th item of K;
  else K-item = i-th item of K;
  P-item = i-th item of P;
  threshold =(distance score of K-item)*α+(numerical score of P-item)*(1- α);
  Let $item_1$ = K-item, $item_2$ = P-item;
  for i = 1 **to** 2 **do**
    score = score of $item_i$
    if (size of top-K Buffer = k) **then**
      and $((score, item_i)$ not in top-K Buffer) **then**
        if (minimum of top-K Buffer) < score **then**
          remove minimum and Add $(score, item_i)$ into top-K Buffer;
    else if ((size of top-K Buffer)< k) **and** $((score, item_i)$ not in top-K Buffer) **then**
          Add $(score, item_i)$ into top-K Buffer;
  end
  if size of top-K Buffer =k **then**
    if minimum of top-K Buffer >= threshold **then**
      return top-K Buffer;
end

4. 실험 결과 및 분석

실험은 Ubuntu 14.04.5 Server 환경에서 python(ver. 3)와 libspatialindex를 진행했다. 데이터는 'Yelp'[i]의 실제 POI 데이터와 임의의 100개의 사용자의 위치정보와 선호도를 10개의 데이터셋으로 진행했다. 실험은 선호도를 직접 추론하는 Dynamic 기법(제안기법)과 고정된 선호도(0.5)를사용하는 Static 기법으로(대조실험) 진행했다[그림4]. 정확도는 Top-K 리스트를 비교하는 기법[1]을 사용해 계산했고, 값이 1에 가까울수록 정확하다는 것을 의미한다. 처리시간은 초 단위로 표시하였다. 최 근접 질의처리를 위한 k'는 50으로 고정하였다. K와 n을 변화하며 진행한 두 실험 모두 전반적으로 제안기법이 정확도가 높은 것을 확인할 수 있다. 특히, k값과 n값이 증가함에 따라 두 기법의 정확도 차이가 더욱 극명해졌다. 처리 시간의 경우 제안기법은 선호도를 추론하므로 대조실험보다 시간이 더 걸리지만, 최대 0.2초차이 정도로 미미한 수준이다.



[그림4] k와 n 실험 결과 그래프

5. 결론 및 향후 계획

사용자가 결정한 우선순위에 따라 시스템에서 선호도를 추론함으로써, 보다 개인화된 결과를 제공하는 시스템을 제안하였다. 전체 질의 시간을 단축하기 위해 선호도 추론에서 사전에 생성된 가상 샘플 데이터를 이용하였다. 결론적으로, 제안 기법을 통해 선호도를 추론하면, 고정된 선호도 값을 적용했을 때보다 더 높은 정확도를 얻을 수 있었다. 추후 연구에서는 다른 속성(평점, 재방문 수치 등)들을 추가하거나, 사용자와 상호작용하는 데 걸리는 시간을 추가로 측정, 혹은 선호도 추론을 위한 상호작용 단계 횟수의 증가 등을 연구할 예정이다.

참고문헌

[1] Kai Zheng, Han Su, Bolong Zheng, Shuo Shang, Jiajie Xu, Jiajun Liu, Xiaofang Zhou, "Interactive Top-k Spatial Keyword queries." *IEEE ICDE*, 01 June 2015

[2] Das, G., Gunopulos, D., Koudas, N., & Tsirogiannis, D. "Answering top-k queries using views." VLDB 2006, September

[3] Min Xie, Laks V. S. Lakshmanan, and Peter T. Wood. "Efficient top-k query answering using cached views." In ICDE EDBT '13). In ACM, 2013. New York, NY, USA, 489-500.

[4] Song M., Park K., Im S., Kong KS. "Nearest Neighbor Queries for R-Trees: Why Not Bottom-Up?" In: Li Lee M., Tan KL., Wuwongse V. (eds) Database Systems for Advanced Applications. Lecture Notes in Computer Science, vol 3882. Springer, Berlin, Heidelberg. DASFAA 2006

[i] https://www.yelp.com/dataset. Subset of Yelp's businesses, reviews, user data for use in personal, educational, and academic purposes. Dataset of 209,393 businesses, reviews by 1,968,703 users.