NETR-Tree: An Efficient Framework for Social-Based Time-Aware Spatial Keyword Query

1 Theoretical Analysis

The searching efficiency depends on the number of candidate nodes. Those searching algorithm evaluating objects according to single criterion sequentially is obviously inefficient. Nevertheless, our algorithm can perform (1) spatial pruning, (2) textual filtering, and (3) temporal check at the same time. Next, we will give an analysis of the time complexity of our algorithm. [1] takes the product of the number of objects accessed and average processing time of single object as the time complexity of their algorithm. We adopt the same estimation method.

Evidently, an object o with high possibility of being a candidate for a query q should satisfy: (1) $o.W \supseteq q.W$, (2) $o.T(q.t) \neq 0$, and (3) $\delta(o.l, q.l) \leq$ the search radius r. For these three requirements, we defined contain-keyword probability $P(o.W \supseteq q.W)$, exist-previous-record probability $P(o.T(q.t) \neq 0)$, and within-radius $P(\delta(o.l, q.l) \leq r)$. Let $P_{can}(o)$ be the probability of the event that o is a candidate and $P_{can}(o) = P(o.W \supseteq q.W) \cdot P(o.T(q.t) \neq 0) \cdot P(\delta(o.l, q.l) \leq r)$. Then we present how to calculate these 3 probabilities respectively.

1.1 Contain-Keyword Probability.

As [2] has pointed out, the keywords in the data set follow a Zipfian distribution. w_i is the word with the i-th highest occurence-frequency. The occurence probability of w_i $P_{oc}(w_i)$ can be calculated through $P_{oc}(w_i) = \frac{i^{-s}}{\sum_{j=1}^{|D.w|} w_j^{-s}}$, where $|D.w|$ is the total number of words in the data set D and s characterizes the skewness. Using the estimation model of Wu et al. [2], the contain-keyword probability is defined as

$$P(o.W \supseteq q.W) = \sum_{any\ list\ of\ o.key} \prod_{j=1}^{|q.W|} \frac{P_{oc}(w_{q.W_j})}{1 - \sum_{k=1}^{j-1} P_{oc}(w_{q.W_k})} \qquad (1)$$

1.2 Exist-Previous-Record Probability.

There exists a pattern in people's daily life that people tend to do something at one certain moment intensively, which follows a normal distribution. Intuitively, the number of check-ins of a object o also follows this pattern. In practice, there are always more than one peak in the distribution. Hence, we use Gaussian Mixture Models. The probability of check-in occurring at a certain moment t follows a superposition of a sequence of normal distributions $N_1(\mu_1, \sigma_1), N_2(\mu_2, \sigma_2), \ldots, N_k(\mu_k, \sigma_k)$. Then, we can get

$$P(o.T(q.t) \neq 0) = \sum_{i=1}^{k} \int_{on\ \tau} f_i(t)dt \tag{2}$$

where $\tau \ni q.t$ and $f_i(t) = \frac{1}{\sqrt{2\pi\sigma_i}} \exp(-\frac{(x-\mu_i)^2}{2\sigma_i})$.

1.3 Within-Radius Probability.

Pruning via spatial constraints is fundamental in spatial keyword query. Chen et al. [1] just assume the query radius is ∞ ignoring the spatial constraints to simplify the computation of time complexity. Here, we give an approximate estimation of within-distance probability $P(\delta(o.l, q.l) \leq r)$. We consider a square whose side-length is $\sqrt{\pi} \cdot r$, thus having the same area as $\odot(q.l, r)$. Leafnodes having intersection with this square have the potential objects that can be candidates. We aim to find the minimum number of leafnodes' MBRs that can cover the square. Let all the leafnodes' MBRs be placed in a rectangular plane coordinate system. The shortest side-length of all MBRs in x-axis direction is defined as l_x, and the shortest distance in the direction of x-axis between any two MBRs that do not overlap in x-axis direction is defined as d_x. We define l_y, d_y in a similar way. Assuming the number of objects contained in a MBR is a constant K, the within-distance probability is defined as

$$P(\delta(o.l, q.l) \leq r) \approx \frac{K}{|D|} \times \lceil \frac{\sqrt{\pi} \cdot r}{l_x + d_x} \rceil \times \lceil \frac{\sqrt{\pi} \cdot r}{l_y + d_y} \rceil \tag{3}$$

With Eqn. (1) (2) (3), we can figure out the value of $P_{can}(o)$. Searching from the root of tree to a leafnode takes $O(\lg |D|)$ time. Assuming the time cost of grading a object o is a constant ζ, the average processing time of single object is $O(\lg |D| + \zeta)$. Hence, the time complexity of our algorithm is

$$O(|D| \cdot P_{can}(o) \cdot (\lg |D|) + \zeta)) \tag{4}$$

References

1. Chen, G., Zhao, J., Gao, Y., Chen, L., Chen, R.: Time-aware boolean spatial keyword queries. IEEE Transactions on Knowledge and Data Engineering (TKDE) **29**(11), 2601–2614 (2017)
2. Wu, D., Yiu, M.L., Cong, G., Jensen, C.S.: Joint top-k spatial keyword query processing. IEEE Trans. Knowl. Data Eng. **24**(10), 1889–1903 (2012)