Advanced Database Project

Nearest Neighbor Search System in Spatial Index Using R-Trees

—

Zhuoran Wu

Department of Computer Science

Georgetown University

<zw118@georgetown.edu>

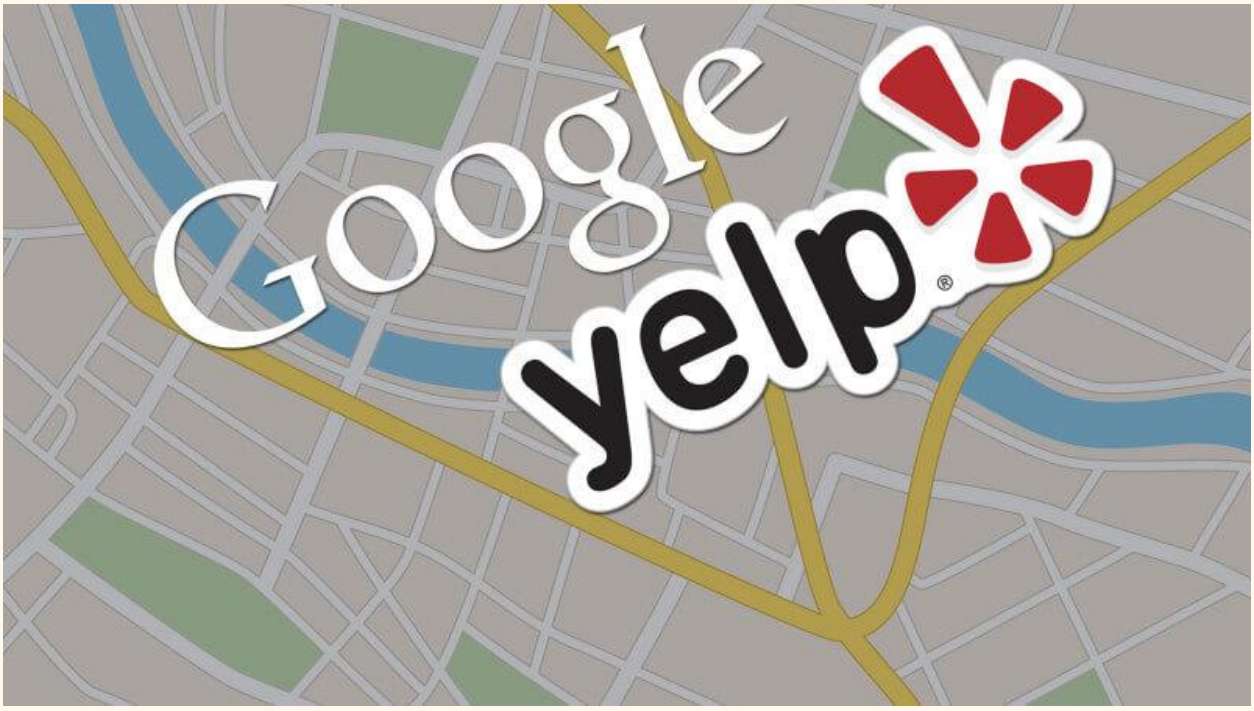


Table of Content

Introduction

The restaurant industry is booming again – by using restaurant marketing with Location-Based Service Apps like Foursquare, Facebook Places, SCVNGR, Google Maps and Yelp. From a user view, the nearest neighbor search in an app is useful to find the closest restaurants, hotels or other businesses within an area.

In spatial data, the application for the search that is able to conduct a nearest search efficiently. In order to conduct the Nearest Neighbor Search(NNS), the DBMS need a spatial structure such as the B-Tree, K-D-B or R-tree to search in.

MySQL current does not support a dedicated NNS algorithm, instead of a full table scan, distance calculations and sorting, NNS is possible. The NNS algorithm should fit into existing structures such as their implemented R-Tree and the way MySQL accesses indexes which are incremental.

In the next few sections, I will introduced about previous work and the model and methods that we used for this project. Also, the Experiment sections will provide the result and analysis for the R-tree index.

Related Work

Nearest Neighbor Search

The Nearest Neighbor (NN) rule is one of the classic pattern recognition problems. The first mentions of this problem is by Alhazen (Ibn al-Haytham, 965-1040 C.E.) one of the scientist of the Islamic golden age. The NN search in proximity searches is an optimization problem of finding the closest point to another point in a set. This paper (J.E.Bery, 2017) provide a detail summary for the NN search.

R-tree

According to Antomn Guttman's original paper about R-tree (Antomn Guttman, 1984), R-tree is the most widely used spatial index structure [3]. The R-tree [4] has the following structure. Suppose B be the disk block size. Then first forms a minimum bounding rectangle (MBR) by grouping B points in the considering area; these grouped points will be stored at a leaf on the R-tree. Further grouping of MBRs has been applied at each level until there exists only one. Each node u in the R-tree is associated with the MBR enclosing all the points stored below, denoted by MBR (u). Each internal node stores the MBRs of all its children.

Methodology

Nearest Neighbor Search

The NN algorithm is a useful technique within data mining that lets you use the data you already know the output value of to create new and unknown output values. This is both similar to regression and classification, but regression can only be used for numerical outputs and that differentiates it from the nearest neighbor immediately. With regards to classification it uses every instance to create a tree, and then traverses that to find the answer, this will become very time consuming for large data sets. Take eBay as an example, to make similar product information would require a very large tree structure; this would be an unreasonable model for this type of data. The nearest neighbor algorithm would solve all these problems and is a very good fit for problems of a similar sort(Abernethy, 2010).

Great-Circle Distance

Which distance calculation one uses to do the NN query is based on the properties of the data. As there are several different distance calculations, to find the best fit for a dataset if of importance. If one does not know the dataset very well, trying out several distance calculations would be preferable. This is mostly when dealing in higher dimensional data for spatial data.

In this project, we have locations data that contains latitude and longitude, so we will use great-circle distance. A great circle is part of a sphere that has a diameter of the sphere. The sections that do not contain a diameter are referred to as small circles. The great circle transfers to a straight line in a gnomonic projection. The shortest distance between two points on a sphere are known as an orthodrome

$$Distance = 6372.8(2\arcsin(\sqrt{\sin(\frac{rad(x_2 - x_1)}{2})^2 + \cos(rad(x_1)) * \cos(rad(x_2)) * \sin(\frac{rad(y_2 - y_1)}{2})^2}))$$

and is part of a great circle (Wolfram, 2017).

Fig 1. Great-Circle Distance Function.

In this function, the distance between two points given the latitude as x and longitude as y.

R-Tree

The R-tree was a proposal by Antonin Guttman in 1984 focused on handling geometrical data, this including points, lines, areas, volumes and other higher dimensional spaces. There are dozens of different implementations of the Rtree, all aimed at solving different problems more efficiently. The R-tree is used within a wide variety of fields such as spatial, temporal, images and multimedia databases. In simple terms the R-tree is a hierarchical data structure based on the BÅ-tree, and is used for the dynamic organization of sets of d-dimensional geometries, by viewing them in their minimum bounding d-dimensional rectangles (MBR) (Mendes et al., 2014).

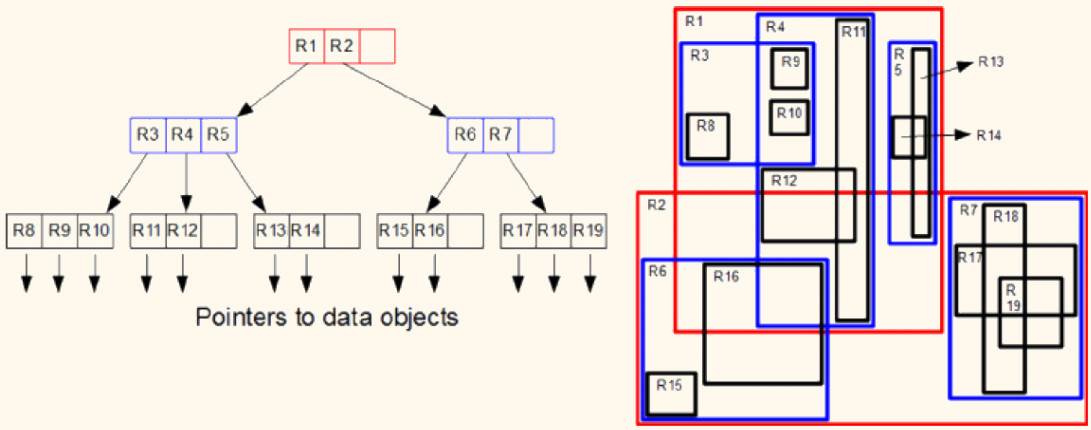


Fig 2. R-tree structure. (Mendes et al., 2014)

Given a R-tree of order (m, M) each leaf node (not the root node) can contain up to M entries, and the minimum allowed number of entries is m <= M/2. The entries are on the form of (mbr, oid). In regards to the internal nodes the maximum capacity is between m <= M/2 and M. The internal nodes entries are on the form of (mbr, p). The root node has a minimal requirement of 2 entries, unless this is also a leaf node, in that case it may contain zero entries or one entry. At last all the leaf nodes are at the same level in the R-tree. (Manolopoulos et al., 2005)

System Structure

In this project, we have 2 parts. The first part is the Website, the second part is the R-Tree part.

In the website part, this time we use Flask [1]as our backend framework. It is a microframework for Python based on Werkzeug[2], Jinja 2[3] and good intentions. The whole website is using Model-View-Controller, MVC model.

[1] Flask: http://flask.pocoo.org/

The model part define the data structures for each business. The result search from dataset will return as a JSON model to the front end.

The view part is our frond-end. We only use CSS + JQuery for front-end processing. All the API call will use Representational state transfer, the RESTful Style.

The controller part, we define several route that flask could process with. Also, in controller, it will call the backend service to get the result.

In order to show the result on the Google Map, this time we use flask_googlemaps[4], which makes easy to use Google Maps in Flask application.

In the R-Tree part, we use two ways to build our R-tree, the first one is to build the R-Tree index based on the current rtree [5]package. This is only for performance evaluation use. The other one is that we build our own R-tree, and test it with the naive method.

Furthermore, we make the R-tree search as a service API to be called by the server. Everytime we get the query request, it will return the query result.

Data

In this application, we use Yelp data released. This dataset is a subset of Yelp's businesses, reviews, and user data. It was originally put together for the Yelp Dataset Challenge which is a chance for students to conduct research or analysis on Yelp's data and share their discoveries. In the dataset we will find information about businesses across 11 metropolitan areas in four countries.

The sample data format for 1 business is like this:

ID	business_id	name	neighborhood	address	city
9	EsMcGiZaQuG1OOvL9iUFug	"Any Given Sundae"	nan	"2612 Brandt School Rd"	Wexford

[2] Werkzeug: http://werkzeug.pocoo.org/
[3] Jinja 2: http://jinja.pocoo.org/docs/2.10/
[4] Flask_Googlemaps: https://github.com/rochacbruno/Flask-GoogleMaps
[5] Rtree: http://toblerity.org/rtree/

state	postal_code	latitude	longitude	stars	categories
PA	15090	40.6151022445	-80.0913487465	5	Coffee & Tea;Ice Cream & Frozen Yogurt;Food

Tab 1. Sample Data for 1 business

For each business, it will have is business_id which is identical, its name, it address with city and state and postal code. Of course, data contains latitude and longitude. Furthermore, it will contain the stars and its categories.

Experiment

Experiment Set-up

The experiment includes in 2 parts. First one we will compare the random directly search according to the locations both from Naive Method and R-tree index. The second is to conduct random Nearest Search according to the locations both from Naive Method and R-tree index.

In order to control the variables in the experiments, in this project, we only measures times based from Python, which means that we will not compare the running time, one from MySQL Engine, the other from python code which need more operations to call the MySQL engine to conduct query.

We have 170,000 rows of business data in MySQL, for each search with latitude and longitude, query need to scan the full table to find the instance. And we make business_id as Primary Key. After we build our R-tree, in each node, it will contain the business information with business_id, latitude and longitude. When we use R-tree, we only need to query in the R-tree, and after we get the result, return the business_id, and make the query in MySQL with primary key. Therefore, the key stuff to compare is the running time of directly query in MySQL in 140,000 rows and the query in R-tree and query based one primary key in MySQL.

About evaluation method, we compare 2 kind of time, Average Time and TotalTime. The less time it used, the better performance that we could tell.

Result and Discussion

First we compare the average running time for Naive method and R-tree method. We conduct queries for 1, 10, 200 and 10000 random search.

Average Time	1 Random Search	10 Random Search	200 Random Search	10000 Random Search
MySQL	0.356752634048461	0.367214719454447	0.366616824282659	0.359334246666
Rtree	0.0665454864501953	0.0465052604675293	0.0473231327533721	0.0496546859264373

Tab 2. Average Time for random NN search

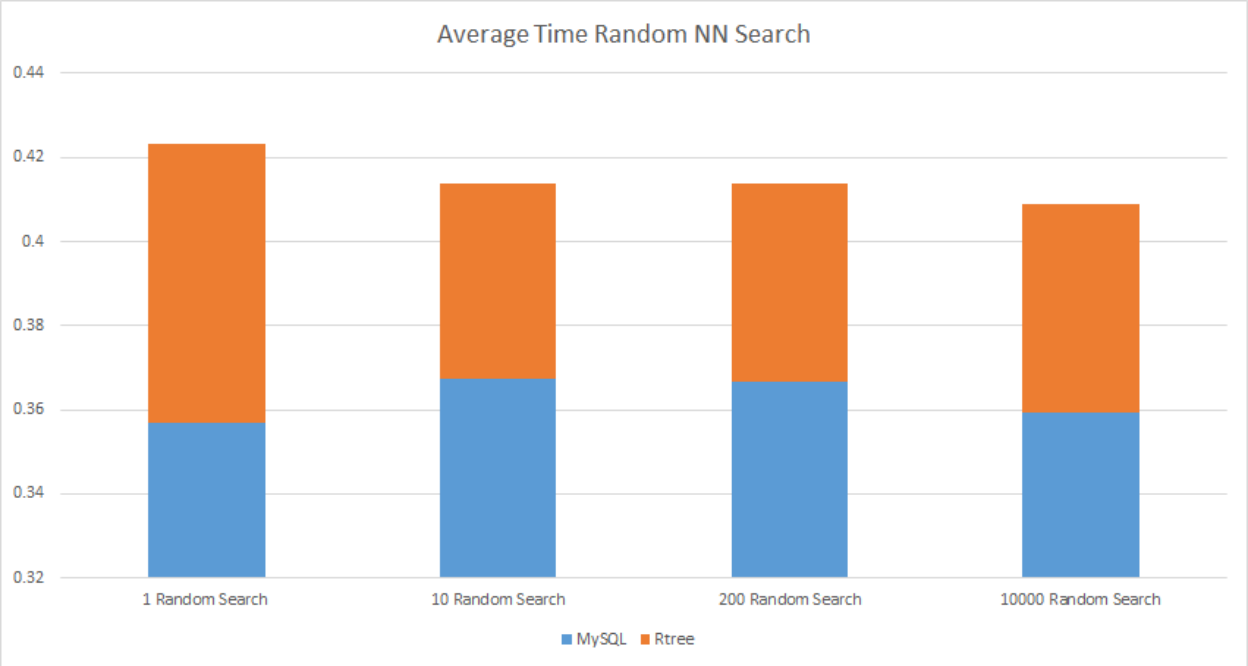


Fig 3. Graph for Average Time for random NN search

According to the result, we found that the query number will not affect the query average time. However, the Naive Method did use more time than the R-tree search. Even if we conduct the 10000 query, the average in two methods will stay almost same.

Then we check total time, time will differ a lot when we conduct the 10000 searches. The total time of Naive method will be more than almost 10 times than the R-tree one.

TotalTime	1 Random Search	10 Random Search	200 Random Search	10000 Random Search
MySQL	0.356752634048461	4.4065766334533 6	81.755551815032 9	3753.546285533
Rtree	0.0665454864501953	0.4644000530242 92	9.5217947959899 9	497.197760820388

Tab 3. TotalTime for random NN search

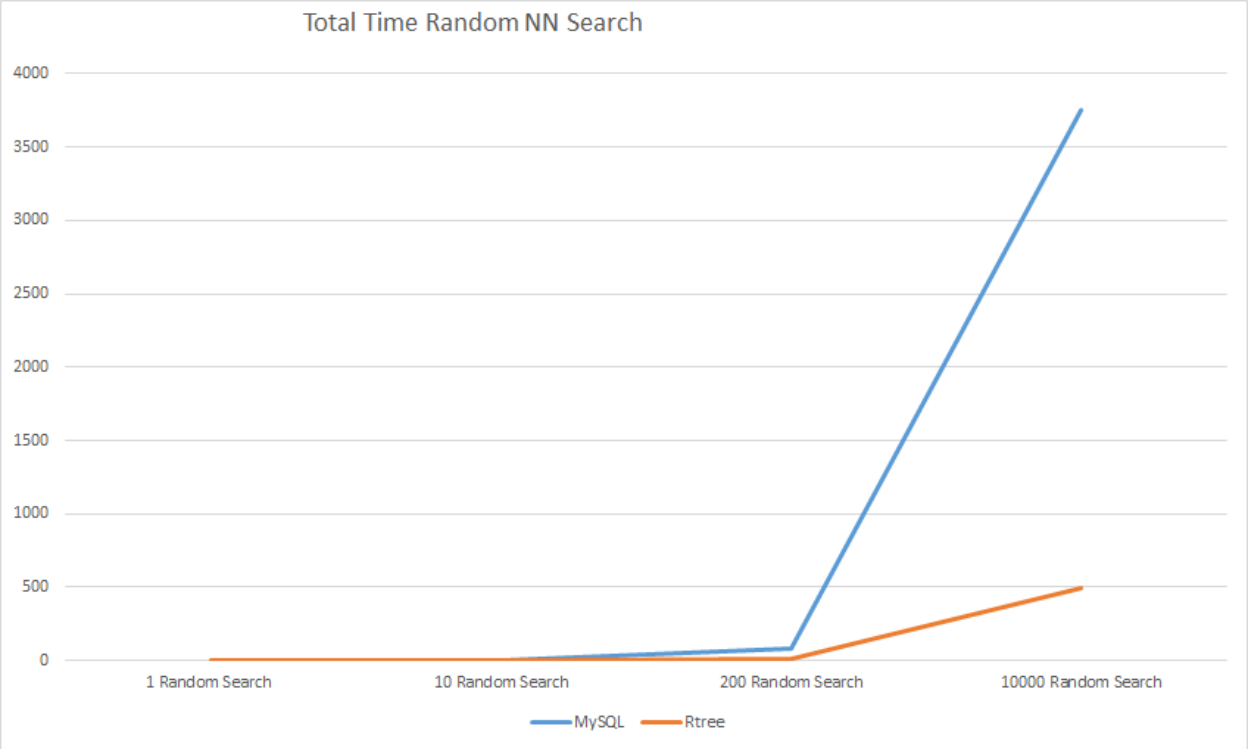


Fig 4. Graph for Total Time for random NN search

Then we compare the result within the R-tree with different Fanout, we found that both in average time and total, the Fanout will not obviously affect the running time. However, it did affect the time for building the tree.

Avg Time	1 Random Search	10 Random Search	200 Random Search	10000 Random Search

Fanout = 10	0.0055036544799804 6	0.0076050281524 6582	0.0094243335723 8769	0.00963548335725 545
Fanout = 25	0.0050063133239746	0.0075042724609 375	0.0085262024402 6184	0.00832620244026 184
Fanout = 50	0.0050034523010253 9	0.0083209443092 3461	0.0083209443092 3461	0.00876094430923 461

Tab 4. Average Time for R-tree search in different Fanout

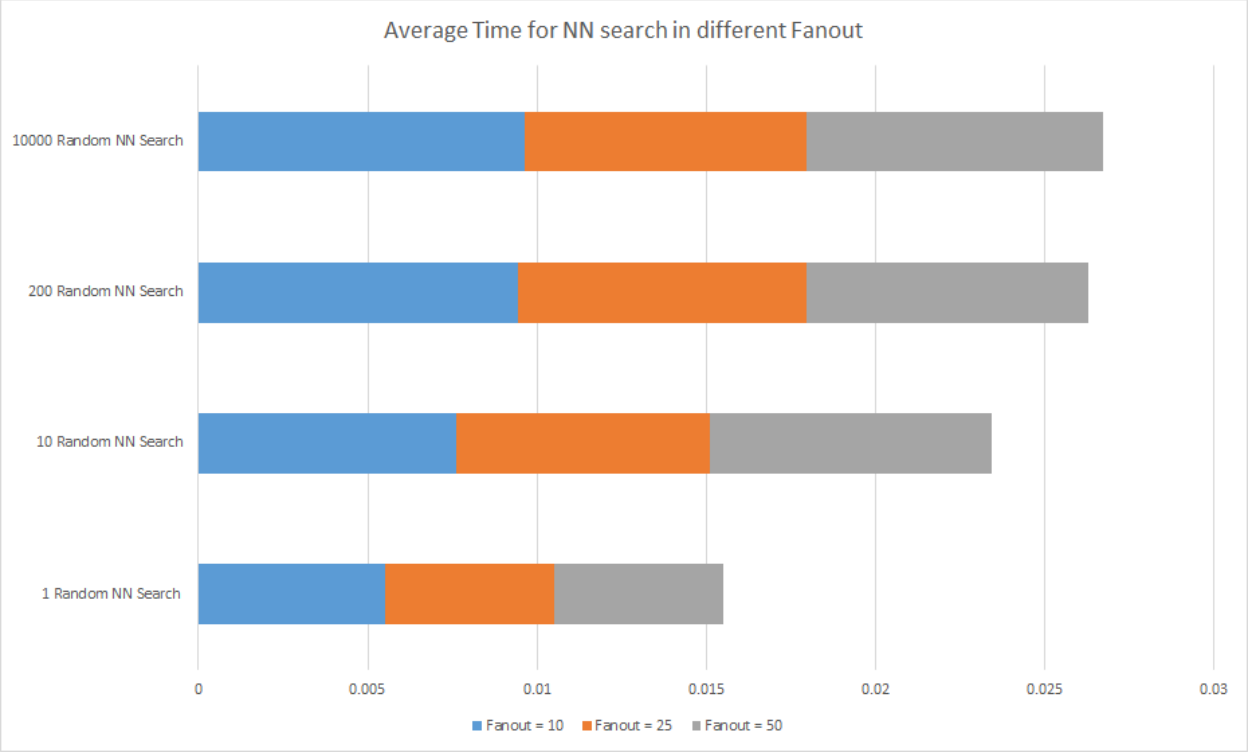


Fig 5. Graph for Average Time for R-tree search in different Fanout

Total Time	1 Random Search	10 Random Search	200 Random Search	10000 Random Search

Fanout = 10	0.0055036544799804 6	0.0760502815246 582	1.8848667144775 3	97.22354355555
Fanout = 25	0.0050063133239746	0.0750427246093 75	1.7052404880523 6	99.75654756556
Fanout = 50	0.0050034523010253 9	0.0675477981567 382	1.6641888618469 2	96.35866557323

Tab 5. TotalTime for R-tree search in different Fanout

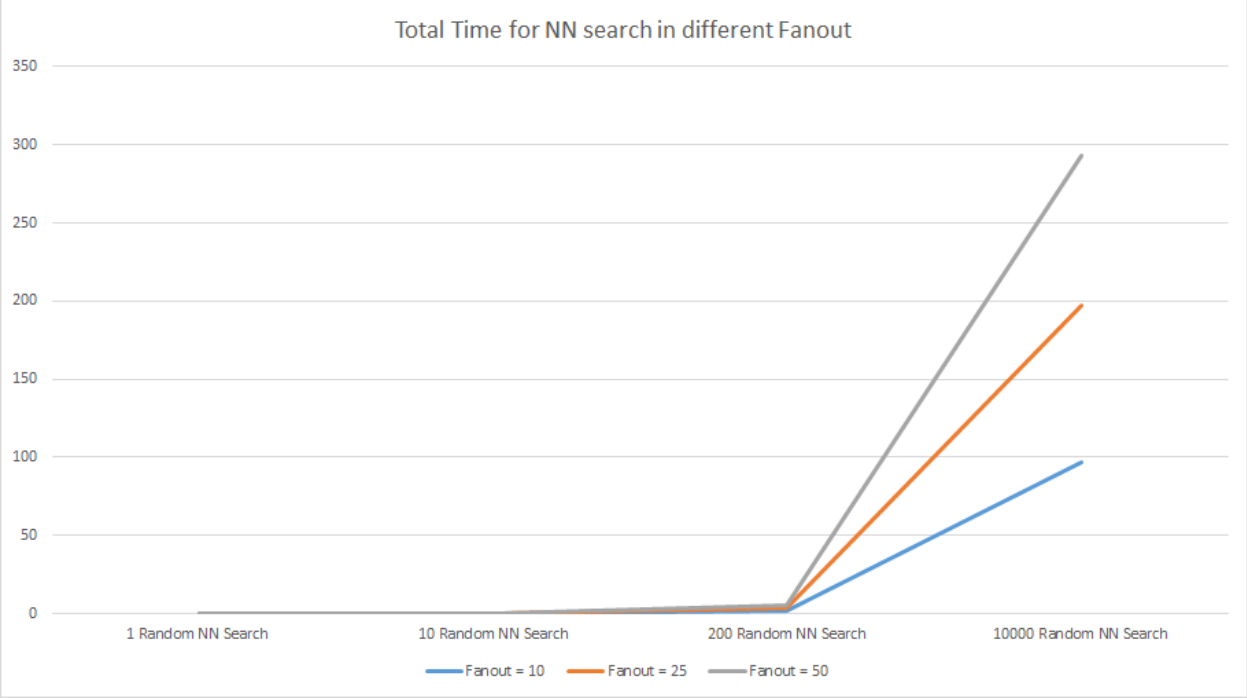


Fig 6. Graph for Total Time for R-tree search in different Fanout

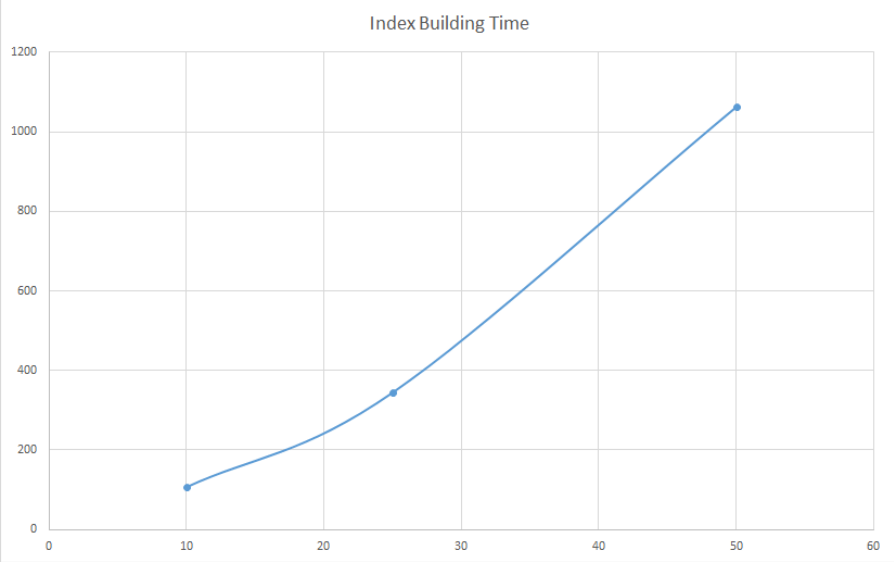


Fig 7. Graph for Index Building Time for different fanout.

Conclusion

From this report, we could see that: 1) R-tree is quite useful in spatial data. R-Tree is has proven to be a very efficient data structure to perform spatial queries. Also, R-tree has made NNS more easily. According to experiment result, it shows that R-tree could significantly decrease the query time for query. Further, its performance is related with the value of fanout. However, in a certain range of fanout, it has no obvious difference.

Future Work

There are still some work that could be conducted for further research. The first one is that the effects on performance for different R-tree types. Also, different parameter setting might influence the performance, this time we test fanout, we could also test the value K of K-means Clustering. Also, different method of pick seed is also considerable.

The second one is that in nearest search, could we just build the index with function in the MySQL directly? In this report, we could only measure the running time. If we could implement the functions in the MySQL, we could evaluation the query time directly.

Reference

[1] Jens Even Berg Blomsøy, Evaluating Algorithms for Nearest Neighbor Searches in Spatial Databases Using R-Trees, Norwegian University of Science and Technology, June 2017.

[2] A. Guttman, “R-trees: a dynamic index structure for spatial searching,” in SIGMOD, 1984.

[3] Nasrin Mazaheri Soudani et al., The spatial nearest neighbor skyline queries, International Journal of Database Management Systems (IJDMS) Vol.3, No.4, November 2011.

[4] Jun Zhang, N. Mamoulis, D. Papadias and Yufei Tao, "All-nearest-neighbors queries in spatial databases," Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004., 2004, pp. 297-306.

[5] Yufei Tao, Aggregate Nearest Neighbor Queries in Spatial Databases, ACM Transactions on Database Systems, Vol. 30, No. 2, June 2005, Pages 529–576.

[6] Jiaheng Lu et al., Reverse Spatial and Textual k Nearest Neighbor Search, Proceeding SIGMOD '11 Proceedings of the 2011 ACM SIGMOD International Conference on Management of data Pages 349-360.

[7] Hideki Satoa, Ryoichi Narita, Efficient Maximum Range Search on Remote Spatial Databases Using k-Nearest Neighbor Queries, Procedia Computer Science, Volume 22, 2013, Pages 836-845.

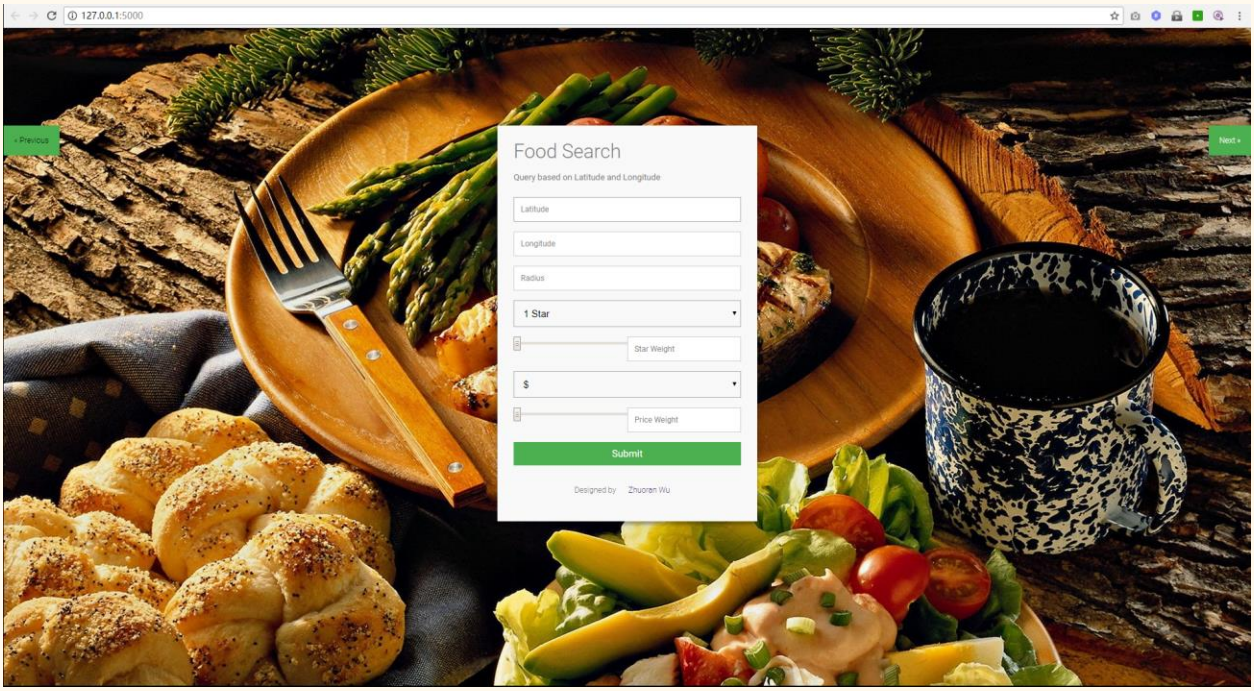
[8] M. L. Yiu, X. Dai, N. Mamoulis, and M. Vaitis, "Top-k Spatial Preference Queries," Proceedings of the 23rd International Conference on Data Engineering (ICDE), Instanbul, Turkey, April 2007.

[9] Mehdi Sharifzadeh and Cyrus Shahabi. 2010. VoR-tree: R-trees with Voronoi diagrams for efficient processing of spatial nearest neighbor queries. Proc. VLDB Endow. 3, 1-2 (September 2010), 1231-1242.

[10] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos. 2004. Algorithms for processing K-closest-pair queries in spatial databases. Data Knowl. Eng. 49, 1 (April 2004), 67-104.

Appendix 1 - Application User Guide

After setting up the website according to README.md instructions. You could see the website is like:



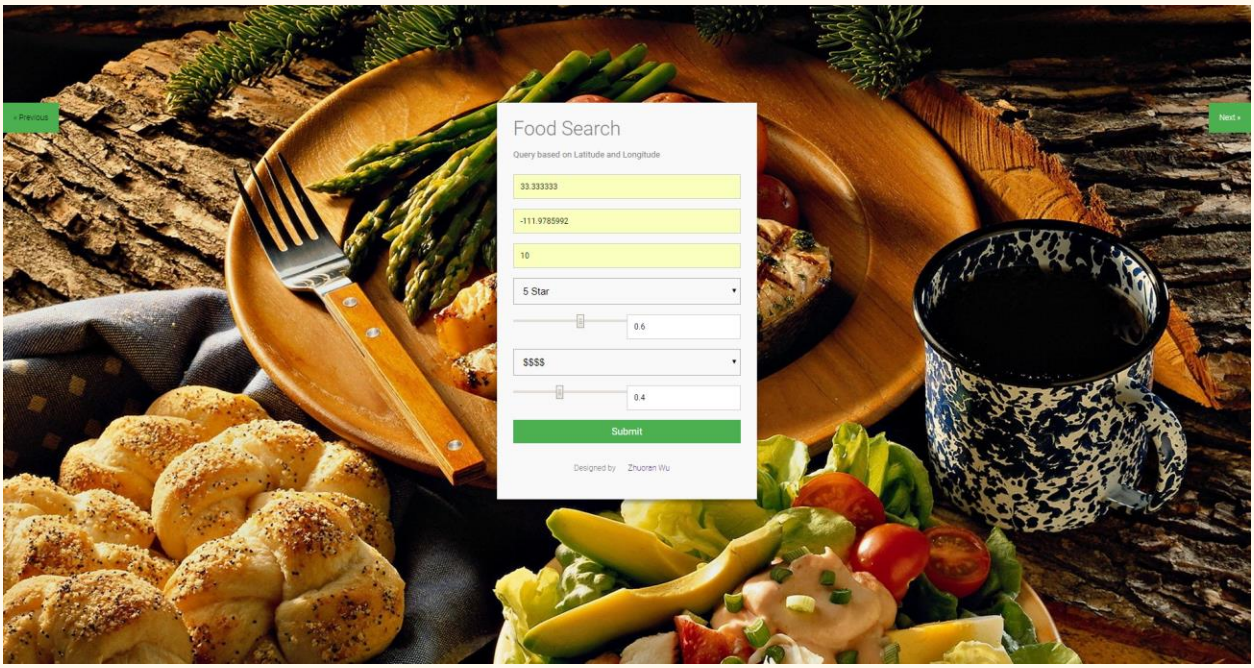
In the first service, you could search business according to a latitude and longitude. For example, we could input the sample point:

Latitude: 33.333333

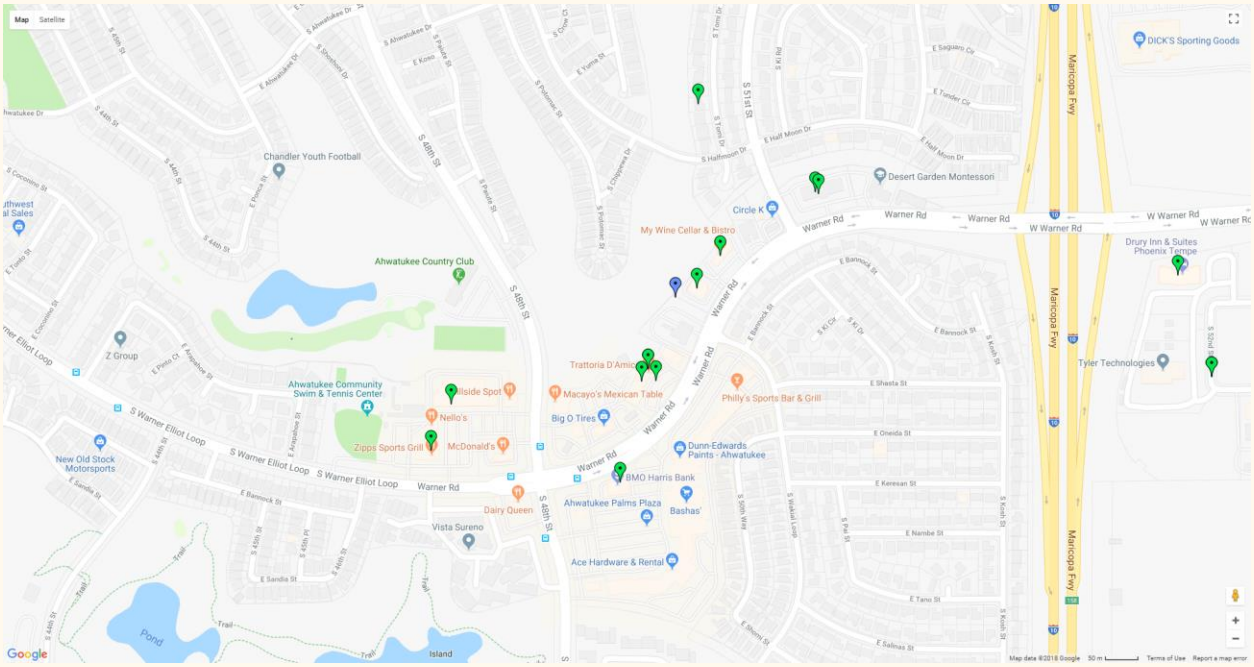
Longitude: -111.9785992

Radius: 20

Also, you could search for the star and price and also assign the weight for both variables.



The result will show in the map. If you click the point, it will show the detail info about the business.



For the Location based service, it has the similar function. However, you only need to input a location, such as:

Location: 5002 Warner Rd Phoenix, AZ 85044