Final Year Project Plan

SpaceKey: Exploring Patterns in Spatial Databases

Supervisor:
Dr. Reynold C. K. Cheng

Team Member:
Yin Yue / 3035234234
Wang Jikun / 3035234715
Lukito Budiman / 3035253864
# Table of Contents

Table of Contents

1. Project Background  
2. Project Objectives  
3. Project Methodology  
   3.1 Data collection  
   3.2 Application of SPM  
      3.2.1 The problem of unwanted object  
      3.2.2 The problem of realistic distance  
      3.2.3 Directional relations  
      3.2.4 Custom objects  
   3.3 Application Details  
      3.3.1 Additional attributes  
      3.3.2 Simple distance constraint specification by learning  
   3.4 Development structure and specs  
4. Project Schedule and Milestones  
5. Division of Labor  
References

1
2
3
5
5
5
5
6
6
6
6
6
7
8
9
10
1. Project Background

With the significant market size of real-estate market across the world [1], the demand for property has reached a new level. At the same time, searching for properties through the internet has become a convenient approach for users. There are several existing property searching applications, such as Airbnb and ziroom, which help users to find their desired property locations.

After investigating on these property searching applications, our observations shows that these applications only supports simple searching functionalities, by searching for single elements and applying simple filters. However, in real life scenarios, users might have more complex requirements on their desired property, for example, the user might want a school as well as a hospital not too close nor too far from his property. As a result, a more advanced searching utility is in demand.

Recently, a new type of spatial group keyword query is raised in researches [2], which is called spatial pattern matching (SPM). Spatial pattern is a graph with its vertices being keywords and its edges being distance constraints between any pair of keywords. This type of query matches a group of objects with their keywords and distance constraints satisfying the spatial pattern specified by the user. Its solutions have been raised and well-discussed.

With the help of SPM and its solutions, we can achieve the advanced searching functionality mentioned above. As a result, our outlook is to realize this kind of advanced searching utility in these housing applications as a module. In this project, our goal is to demonstrate the feasibility and importance of this searching utility by developing a standalone web-based application.

The remaining parts of this project plan were arranged as the followings. First, we described the objectives of our project in detail. Then it followed by the detail implementation of our products, the web application. Lastly, the project timeline and milestones are presented.
2. Project Objectives

Our outlook is to integrate our project as a module with existing housing applications to do advanced house searching/filtering with the help of SPM. In this module, users can filter available housing locations according to their interest in surrounding keyworded points of interests (PoI), such as supermarkets, schools, etc. They can specify distance and directional constraints between the desired property and keyworded/user-specified locations by drag-and-drop actions and clicks. At the same time, it will provide more information related to the locations to the user, such as 24-hour traffic, reviews, ratings, etc.

In this project, the main goal is to make a standalone demo application that shows how the advanced house searching/filtering works, which includes all the above functionalities.

The objectives include:

1. A set of supporting open-source API that supports SPM queries.
   
   The first objective is to build a set of supporting APIs for the project. It will focus on the spatial query algorithms, specifically SPM, as well as many other existing algorithms, such as topK, mCK, minSK, CoSKQ.

   It will achieve following basic functionalities:
   - Input data from local sources
   - Build various data structures to support corresponding spatial keyword query types
   - Answer various spatial keyword queries

   The API set will be open-source, so anyone can use it to build new applications in the future.

2. A set of sample data for the demo
   
   The next objective is to obtain a demo dataset for the demo application. The dataset need to include the following things:
   - Housing locations
     - Spatial Information: latitude, longitude
     - Housing Information: price, house size, etc.
   - Points of Interest (PoI) locations
     - Spatial Information: latitude, longitude
     - Keyword Information: A set of feature keywords for each object
     - Additional Information: ratings, reviews, etc.
   - Distance information
     - Realistic distance: travel time, 24-hour traffic, etc.

   Note that the sample data could either be authentic data or generated data, depending on our approach.

3. A standalone demo housing application
   
   A standalone demo housing application will be made to demonstrate the industrial value of this project and realize its outlook. The demo will be a web-based application. An android version will also be developed as an optional addition.
It will have following basic functionalities:

● Pattern Specification

The demo application will feature a easy-to-use and advanced pattern input interface. It will allow users to input the spatial pattern by dragging icons by cursor on PC or by hand on handheld devices and specify distance and direction requirements by clicking/touching. The distance constraints specification will be straight-forward, with options such as near, far, etc. It will also allow users to specify other requirements for the location, such as rush-hour traffic, etc. In addition, other functionalities and options will be provided, such as inputting custom objects, etc.

● Query processing

The demo application will be able to process the query utilizing the supporting API mentioned above and return the results in a considerably short amount of time.

● Results display

The demo application will feature a highly comprehensible result display based on a map. It will display all the results that satisfy the requirements for the user, one at a time. The user can look at other result by clicking buttons. Additional information regarding the locations will also be displayed if needed, such as traffic information, reviews, ratings, etc.

● Additional functionalities/queries

The demo application will also enable other types of spatial keyword queries for comparison purpose.

4. A project website

A website will be developed to display information related to the project

It will include the following contents:

● The documentation of the usage of the supporting API set.

● The documentation of the usage of the demo housing application.
3. Project Methodology

3.1 Data Collection

Due to the limitations of SPM’s algorithms, map data needs to be local during searching process, so data collection would be the first step along the implementation.

The required data is related to map and housing, which can be obtained through data crawling on some websites that provide corresponding services, such as Google Maps, etc.

The implementation of scraper would rely on some Python libraries like selenium, beautifulsoup4, scrapy, etc. The crawling approach is based on script-controlled HTTP request. It obtains the data by parsing the corresponding HTTP responses. Library like selenium can simulate user interactions with the browser to skip anti-crawling barriers on some of the websites. After the raw data is gathered, it needs to be formatted as elements for algorithmic usage, and they would be categorized as properties and map elements.

External APIs like Google Maps API can also be utilized to extract realistic distances that are measured in terms of travel time from the property to the destination. Due to the resource limitations, we decided to use a free trial account. However, with this account, obtaining the pairwise distances of 500 points would take 1 month, and 4 months is required for that of 1000 points, and so on. Therefore, the data that can be obtained from Google Maps API would be limited. In other words, if the application requires only the data within certain region/country, or we use the data in an alternative way, then the scraper might be an option.

The complete map would be the domain of the algorithm to achieve spatial pattern matching and it would be temporarily stored in the storage of the API module. The scraper will be executed periodically and independently to update and import data to the map database.

3.2 Application of SPM

3.2.1 The Problem of Unwanted Objects

In SPM, when a keyword is specified in the pattern, the returned result must contain that keyword regardless of the magnitude of the distance constraints. In real world scenario, when a user just want any instance of a keyword to be far enough from the location, such as airport, the user won’t care about any instance of that keyword. Furthermore, matching an instance that is far away will hurt the performance of the query. Therefore, we propose three different approaches to deal with this problem.

Suppose the query pattern is P, the set of unwanted keywords is U.

1. Directly use the SPM algorithm on P, setting the interval upper-bound with the unwanted object to a large number (+inf)
2. Use the SPM on P - U, and filter the result set so that no unwanted keywords in U is within the unacceptable distance of the corresponding keywords.
3. Use the SPM on P - U, get result set R. Then for each keyword u_i in U, perform SPM on P - U + {u_i}, get result set R_i. Then the final result will be R - SUM{R_i}.

More solutions will be proposed and investigated. The performance will be evaluated and compared to obtain the optimal solution.
3.2.2 The Problem of Realistic Distance

In SPM, the distance is measured by Euclidean distance between locations, which is not very realistic: locations that are geometrically closer to each other may take more time to travel between. Therefore, more realistic distance measurement is needed in this application, such as travel time, route distance, etc. Since the algorithm requires distance data to be local, we need to obtain route data instead of invoking existing APIs, such as Google Maps API. Due to the route data source limitations mentioned above, in the case of no other data source is available, we propose alternative ways to utilize these data: Euclidean distance will be used in the algorithm, with looser distance intervals decided by the app’s specification. Then, the routing queries on some map API will be used to get the realistic distance only on the result set to filter/display them afterwards.

3.2.3 Directional Relations

In SPM, no directional relations could be specified in a pattern. However, in a real world scenario, a user might want to specify directional constraints between objects, e.g. a school to be on the northwest of the housing location. This functionality can be easily achieved by applying a filter on the result set to validate directional relation.

3.2.4 Custom Objects

We want to enable user with the ability of specifying custom objects in the pattern. It could be very useful if the user has some locations in his mind to relate to, e.g. workplace, friends’ house, etc. In our project, the user will be able to specify a custom object by clicking on the map or specify latitude and longitude. We can achieve this by assigning a unique keyword to the location that the user specifies temporarily, so that the result must include the location with that keyword. In addition, more useful information could be displayed related to the custom objects. In the case of workplace or friends’ house, 24-hour traffic information (especially rush hours) will be very useful for the user. This information could be obtained from various existing map APIs.

3.3 Application Details

3.3.1 Additional Attributes

In our demo, we want to add the functionality to display other information related to the housing location and PoIs, which is helpful for the user to make the decision. The additional attributes include ratings, the number of reviews, user review, opening hours, etc. Also, we can use some attributes to decide the displaying order of the results, as well as filtering them, such as ratings and the number of reviews. These data can be pulled from various existing map APIs.

3.3.2 Simple Distance Constraint Specification by Learning

In this project, users can specify the distance constraint using a simple version (e.g. near, medium, far) and the numerical version (e.g. 1km, 500m). To obtain the mapping from the simple version to the numerical version, collection of user’s preferences would be required. Objects with different keywords may differ in terms of this kind of mapping (e.g. near in terms of airport and supermarket).

The approach is to record and store user’s preferences to learn the appropriate mapping related to a specific keyword. As a result, the usage of customizing numerical distance constraints should decline along the way, meaning that the mapping has a reasonable estimation of the numerical distance.

The concept is to improve user-experience through crowdsourcing related to distance measurement. Moreover, if an existing open-source crowdsourcing learning library can be found, it is possible to directly utilize it without further investigation into the methods.
3.4 Development Structure and Specs

![System Architecture Diagram](image)

Figure 3. The system architecture

**API set:**

The supporting API set will be developed using Java. It will access the database via JDBC. It will be later hosted on either one of the cloud services such as AWS or on our own dedicated machine.

**Database:**

The database will be developed and maintained in MySQL. It will be later hosted on either one of the cloud services such as AWS or on our own dedicated machine.

**Web application:**

- Front-end part will be built using HTML, CSS, and JavaScript, as well as some front-end framework such as Bootstrap.
- For the back-end part, we will use Spring framework.

**Mobile application:**

- The front-end part will be developed using Java.
- For the back-end part, it will share the same back-end server with the website to increase the reusability of the server-side. Universal protocol would be defined through tools like Protocol Buffers to simplify the communication procedure between client and server.
# 4. Project Schedule and Milestones

<table>
<thead>
<tr>
<th>Date</th>
<th>Milestones</th>
</tr>
</thead>
<tbody>
<tr>
<td>30/9/2018</td>
<td>Deliverables of Phase 1 (Inception)&lt;br&gt;● Detailed project plan&lt;br&gt;● Project web page</td>
</tr>
<tr>
<td>30/09/2018 - 30/10/2018</td>
<td>● Create a basic website interface&lt;br&gt;● Research about collecting data set as well as generating data&lt;br&gt;● Develop the spatial keyword query API</td>
</tr>
<tr>
<td>30/10/2018 - 30/11/2018</td>
<td>● The website is able to do basic query and advanced query&lt;br&gt;● Make a more detailed version of the website&lt;br&gt;● Collect and generate the demo dataset&lt;br&gt;● Set up the database&lt;br&gt;● Customize the spatial keyword query API to fit this project&lt;br&gt;● Do code review</td>
</tr>
<tr>
<td>30/11/2018 - 06/01/2019</td>
<td>● Add user-register capability, single-sign-on (SSO) capability, and user messaging feature to the website&lt;br&gt;● Automate data collection&lt;br&gt;● Do black-box testing&lt;br&gt;● Do performance optimization&lt;br&gt;● Prepare for the first presentation and detailed interim report.</td>
</tr>
<tr>
<td>7/01/2019 - 11/01/2019</td>
<td>First presentation</td>
</tr>
<tr>
<td>20/01/2019</td>
<td>Deliverables of Phase 2 (Elaboration)&lt;br&gt;● Preliminary implementation&lt;br&gt;● Detailed interim report</td>
</tr>
<tr>
<td>11/01/2019 - 10/02/2019</td>
<td>● Do User Acceptance Testing (UAT) on different browser and platforms&lt;br&gt;● Create a basic android application interface</td>
</tr>
<tr>
<td>10/02/2019 - 20/03/2019</td>
<td>● The mobile version is as functional as the website version</td>
</tr>
<tr>
<td>20/03/2019 - 13/04/2019</td>
<td>● Do final testing, fix mistakes&lt;br&gt;● Prepare for final report, final presentation, and project exhibition.</td>
</tr>
<tr>
<td>14/04/2019</td>
<td>Deliverable of Phase 3 (Construction)&lt;br&gt;● Finalized tested implementation&lt;br&gt;● Final report</td>
</tr>
<tr>
<td>15/04/2019 - 19/04/2019</td>
<td>Final presentation</td>
</tr>
<tr>
<td>29/04/2019</td>
<td>Project exhibition</td>
</tr>
<tr>
<td>29/05/2019</td>
<td>Project competition</td>
</tr>
</tbody>
</table>
5. Division of Labor

<table>
<thead>
<tr>
<th>Task</th>
<th>Assigned to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data collection</td>
<td>Yin Yue</td>
</tr>
<tr>
<td>API</td>
<td>Jikun</td>
</tr>
<tr>
<td>Web application front-end</td>
<td>Budiman</td>
</tr>
<tr>
<td>Back-end</td>
<td>Yin Yue, Jikun, Budiman</td>
</tr>
<tr>
<td>Mobile application front-end</td>
<td>Yin Yue</td>
</tr>
<tr>
<td>Database administrative</td>
<td>Jikun</td>
</tr>
<tr>
<td>System testing</td>
<td>Budiman</td>
</tr>
<tr>
<td>UAT</td>
<td>Yin Yue, Jikun, Budiman</td>
</tr>
<tr>
<td>Project report, presentation and exhibition</td>
<td>Yin Yue, Jikun, Budiman</td>
</tr>
</tbody>
</table>
References
