A Proposal of Spatial Operators for a Collaborative Map Search System

Yuanyuan Wang
y.wang@yamaguchi-u.ac.jp
Yamaguchi University

Panote Siriaraya
spanote@gmail.com
Kyoto Sangyo University

Yukiko Kawai
kawai@cc.kyoto-su.ac.jp
Kyoto Sangyo University/ Osaka University

Keishi Tajima
tajima@i.kyoto-u.ac.jp
Kyoto University

ABSTRACT

We proposed a new query language that expresses complex spatial queries in a concise and intuitive way. The language can express conditions on the range, direction and time distance within a spatial search query by using the proposed spatial operators and "space character". We also show how a collaborative map search system supporting this query language can be implemented and describe several applications to highlight how the language can be put into practice.

CCS CONCEPTS

• **Information systems** → *Query languages*; *Information retrieval query processing*; Query reformulation.

KEYWORDS

query language, spatial search, spatial operators

ACM Reference Format:
Yuanyuan Wang, Panote Siriaraya, Yukiko Kawai, and Keishi Tajima. 2019. A Proposal of Spatial Operators for a Collaborative Map Search System. In *24th International Conference on Intelligent User Interfaces (IUI '19 Companion), March 17–20, 2019, Marina del Rey, CA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3308557.3308703

1 INTRODUCTION

Most map search systems are designed so that general users can easily search for locations and places by using simple keyword-based queries. Such queries are however inadequate when more complex requests are needed. For example, consider a software tool to help users identify appropriate locations for real estate development. Such software needs to deal with search requests which contain multiple constraints and spatial conditions. For instance, those interested in a family apartment would need to search for vacant locations that have many schools and playgrounds within traveling distance but are far enough away from inappropriate locations or noisy public spaces.

In map search systems with simple keyword-based queries, such a search task requires multiple query transactions by user operations. Proximity operators are used as a way to limit searched documents to those that certain keywords which are of a specified order or distance [2]. Oracle Text utilizes a proximity operator (near) which requires the query keywords to appear within a default number of word distances [3]. For example, an operator such as 'near((dog, cat), 6)' could be used to search for documents which contain the word 'dog' which is within 6-word lengths of the word 'cat'. In regards to finding appropriate routes, there have been many studies aimed at helping users locate simple and memorable routes [1, 4], For example, to search for a school around a new house, the transactions need to (1) identify schools on the map by using a keyword query, (2) limit the query result to those within 4km from the found schools, and then (3) exclude locations which are within 200m from each of the found inappropriate locations. The same is true for other complex spatial based requests such as "Finding all the restaurants located between two famous tourist landmarks" (when developing an application for tourism) or "Finding all the shops located next to parks in a city" (for location-based recommender systems). As shown in these examples, it is difficult to process complex search requests using simple keyword-based queries. Systems only supporting such keyword-based queries require multiple steps including non-textual interactions to process them.

On the other hand, there has been much research on spatial logic or algebra. In a spatial database of PostGIS , the spatial search task with location queries can be run in SQL and they can represent complex spatial conditions and queries, but are too complicated for general users in many applications and therefore impractical for the use in such systems.

The goal of this research is to design a simple spatial query language which can represent complex queries within a single query statement with a concise and intuitive syntax so that users can easily specify complex queries. Our language uses various spatial operators which allow users to incorporate spatial conditions and manipulate spatial areas within their keyword-based-like queries.

2 THE SPATIAL QUERY LANGUAGE

A description of our proposed spatial query language is described in this section. Overall, there are three main rules which are used to define the syntax for a spatial search unit in our query language.

Rule 1: The syntax of the most primitive unit of spatial queries is defined as follows: "A␣distance␣α" A and α are keywords, with A denoting the location of the origin for the spatial search

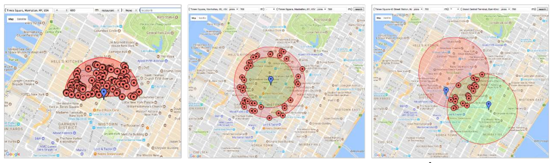
(a) ^ operation (b) - operation (c) * operation

Figure 1: Examples of map search using spatial operators.

Table 1: Spatial operators by using the space-key

Operator	␣*	␣^	␣@	␣[x-y]
Processing	Surrounding	Direction (north/up)	Angle	Range

for the object α. When continuous space is used, the distance would represent the N nearest locations with the property α, where N is represented by the number of continuous spaces.

Example : A␣800m␣α denotes a query statement to identify the α objects which exist inside the region 800m from the origin point A.

Example : A␣␣␣α denotes a query statement to identify the nearest 3 α objects from the origin point A.

Rule 2: The keywords (e.g., A and α) used in the primitive unit of the spatial query would be encapsulated within a double quotation mark (e.g., "Tokyo tower" or "Grand Central, New York" for A or "pizza shop" or for α).

Rule 3: Each primitive spatial search Unit could be combined with other units through the use of spatial, directional and distance operators in a mathematical equation format (e.g., (A␣800m␣α) + (B ␣300m␣α)).

Our language introduces 10 operators to represent conditions in directions, ranges, angles, and time distance. Figure 1 shows examples of map search carried out by queries using spatial operators, e.g., ([^] [-] [*]) in our language (through a query such as ("my house"␣0.5km␣"pizza") * ("friends house"␣ 0.5km␣"pizza") to provide the results of Figure 1 (c)).

2.1 Spatial Operators

The standard set operators can also be used as spatial operations for the query unit defined previously. These include the union [+], difference [-], intersection [*] (Figures 1, 2).

2.2 Directional and Distance Operators

Table 1 shows several spatial operators which are used to further denote distance and direction within our spatial query.

2.3 Range, Size, and Time Operators

Our proposed spatial query language also includes a variety of range operators which allows users to more accurately specify the desired search range within their query. Size [$] and time operators [#] could be used to formulate search queries. The size operator $ extracts the size of the corresponding property of object α and uses it as a unit of measure (i.e., 1 city block = 0.5km).

3 SPATIAL SEARCH SYSTEM

To highlight how the system could be useful in practice, a number of web applications were created which utilized our proposed spatial search query language. The first application was a web interface for our search system which users could use to test the query language

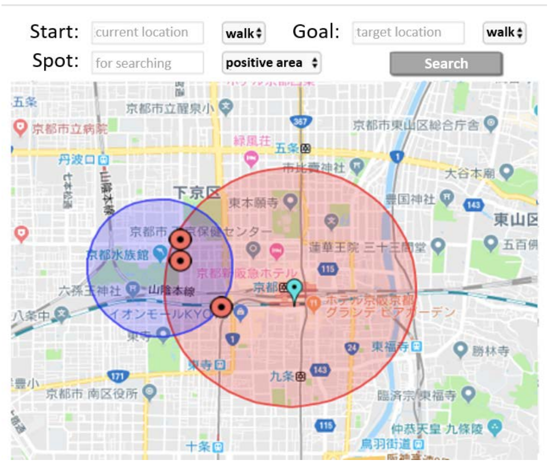

Figure 2: Collaborative map search system.

or search for locations using spatial equations[1]. Users can use the spatial, range and directional operations as well as mathematical expressions such as brackets to compose their search queries. After clicking the search button, the system would send the user's query to the search system server and would then render the search results received from the server onto the map. Furthermore, an application which utilized only the "space-key") for a more simple and intuitive search was also developed [2]. Figure 2 shows another demonstration application which was developed using our API as a "meet up application". This application would identify potential meeting places for three users based on their starting locations (the latitude and longitude of which could be acquired from a mobile terminal). The screen shot in Figure 2, shows an example when one user works near Kyoto Station and the other near Aqua Museum and the system would need to find a restaurant that is equally near to all of their workplaces for them to meet for lunch. In addition, it is also possible to set an avoidance area when users wish to avoid an encounter.

4 CONCLUSION

In this paper, we proposed a new spatial query language for spatial search which can be used to express complex search queries in a text equation format. We implemented a prototype of our proposed system and developed applications such as a collaborative map search system "meet up application".

5 ACKNOWLEDGMENTS

The work in this paper is partially supported by JSPS KAKENHI Grant Numbers 16H01722, 15K00162, 17K12686.

REFERENCES

[1] M. Duckham and L. Kulik. 2003. "Simplest" Paths: Automated Route Selection for Navigation. In *Proc. of COSIT2003*. 169–185.
[2] Elsevier Newsletter. 2015. How Can I Search Literature with Reduced Noise? Utilization of "Proximity Operator" in ScienceDirect & Scopus.
[3] Oracle Text Reference. 2015. 11*g* Release 2 (11.2), B61357-06. https://docs.oracle.com/cd/E16338_01/text.112/b61357/title.htm.
[4] S. Wakamiya, H. Kawasaki, Y. Kawai, J. Adam, E. Aramaki, and T. Akiyama. 2016. Lets Not Stare at Smartphones While Walking: Memorable Route Recommendation by Detecting Effective Landmarks. In *Proc. of UbiComp2016*. 1136–1146.

[1]http://yklab.kyoto-su.ac.jp/~sakata/spatialQueryDemo
[2]http://yklab.kyoto-su.ac.jp/~ichimura/spatialQuery