SWORS: A System for the Efficient Retrieval of Relevant Spatial Web Objects

Xin Cao[§] Gao Cong[§] Christian S. Jensen[†] Jun Jie Ng[§]
Beng Chin Ooi[‡] Nhan-Tue Phan[§] Dingming Wu[♯]

[§] Nanyang Technological University, Singapore
[†] Arhus University, Denmark
[‡] National University of Singapore, Singapore
[♯] Hongkong Baptist University, Hong Kong
[§] {xcao@e., gaocong@, ngju0017@e., phan0032@e.}ntu.edu.sg, [†]csj@cs.au.dk,
[‡] ooibc@comp.nus.edu.sg, [♯] dmwu@comp.hkbu.edu.hk

ABSTRACT

Spatial web objects that possess both a geographical location and a textual description are gaining in prevalence. This gives prominence to spatial keyword queries that exploit both location and textual arguments. Such queries are used in many web services such as yellow pages and maps services.

We present SWORS, the Spatial Web Object Retrieval System, that is capable of efficiently retrieving spatial web objects that satisfy spatial keyword queries. Specifically, SWORS supports two types of queries: a) the location-aware top-k text retrieval (LkT) query that retrieves k individual spatial web objects taking into account query location proximity and text relevancy; b) the spatial keyword group (SKG) query that retrieves a group of objects that cover the query keywords and are nearest to the query location and have the shortest inter-object distances. SWORS provides browser-based interfaces for desktop and laptop computers and provides a client application for mobile devices. The interfaces and the client enable users to formulate queries and view the query results on a map. The server side stores the data and processes the queries. We use three real-life data sets to demonstrate the functionality and performance of SWORS.

1. INTRODUCTION

With the proliferation of geo-positioning, e.g., by means of GPS or systems that exploit the wireless communication infrastructure, accurate user location is increasingly available. On the one hand, many geo-referenced objects are being associated with descriptive text documents (e.g., geo-locations in Google Maps have textual descriptions). Next, on the other hand, increasing numbers of web documents are being geo-tagged (e.g., Google Places enables the association of location with the web sites of businesses). Therefore, very large and growing amounts of objects are available on the web that have an associated geographical location and textual description. Such *spatial web objects* include stores, tourist attractions, restaurants, hotels, and businesses.

This gives prominence to *spatial keyword queries* [1–5] (References [1, 2] provide additional references). Such a queries generally take a location and a set of keywords as arguments and retrieve spatial web objects that best match the arguments. The location component represents a specific local intent, and the keywords component describes user preferences.

Some forms of spatial keyword queries are supported in real-world applications, such as yellow pages and Google Maps. For example, users can specify an address and a set of keywords in yellow pages services, upon which a list of businesses whose descriptions contain the keywords are returned, ordered by their distances to the specified address. However, the algorithms and data structures used in such applications are proprietary and not disclosed. The text retrieval toolkit Lucene also supports a type of spatial keyword query, namely the one that retrieves objects in a given region that each contains all query keywords (i.e., treating keywords as Boolean filters).

Indeed, most previous research on spatial keyword queries treats keywords as Boolean filters and either finds the objects in a given spatial region that contain the query keywords or retrieves a list of objects containing the query keywords ranked based on their distances to the query point.

In contrast, SWORS supports the *location-aware top-k text retrieval* (LkT) query [2] that computes the text relevance between the descriptions of the objects and the query and then uses both the textual relevance and the spatial proximity for ranking. As an example, a tourist traveling with a child in Paris wants to find hotels that are child-friendly and close to some specific tourist attractions. The tourist might issue a spatial keyword query with the keyword argument "cheap and family friendly hotel." and a location argument that is obtained from the GPS receiver of the tourist's mobile device.

Next, user needs may exist that are not easily satisfied by a single object, but where groups of objects may combine to meet the user needs. Put differently, the objects in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 38th International Conference on Very Large Data Bases, August 27th - 31st 2012, Istanbul, Turkey.
Proceedings of the VLDB Endowment, Vol. 5, No. 12
Copyright 2012 VLDB Endowment 2150-8097/12/08... $ 10.00.

a group collectively meet the user needs. For example, the tourist may have particular shopping, dining, and accommodation needs that are best met by several spatial web objects. To address the need for such collective spatial keyword queries, SWORS also supports the *spatial group keyword* (SKG) query [1]. Such a query retrieves a group of spatial web objects that collectively meet the user needs given as a location and a set of keywords: the textual description of the group of objects must cover the query keywords, the objects must be close to the query point, and the objects in the group must be close to each other.

SWORS exploits the IR-tree index [2], which targets the efficient processing of spatial keyword queries, to process the *location-aware top-k text retrieval* (LkT) queries [2] as well as the *spatial group keyword* (SKG) queries [1].

This demonstration enables participants to experience the functionality and performance of SWORS. Browser-based user interfaces are available for conventional desktop and laptop computers. We also provide a client for mobile devices. They enable users to formulate their queries and view the relevant objects using Google Maps. On the server side, we organize the data using the IR-tree and process the queries utilizing this index. Queries are sent from the browser or the client to the server by the standard HTTP post operation.

The rest of the demonstration proposal is organized as follows. Section 2 describes the data model and provides the formal definitions of the queries. Section 3 presents the architecture and design of SWORS. Finally, Section 4 offers the demonstration details.

2. DATA MODEL AND QUERIES

We present the data model used in SWORS and give the formal definitions of the queries supported in SWORS.

Let S be a universal set of keywords. The keywords capture user preferences. Let D be a database consisting of spatial web objects. Each object o in D is associated with a location $o.\lambda$ and a set of keywords $o.\psi$ ($o.\psi \subset S$) that describe the object (e.g., the menu of a restaurant).

2.1 Location-Aware Top-k Text Query

A location-aware top-k text retrieval (LkT) query $q = \langle q.\lambda, q.\psi \rangle$ has a location descriptor $q.\lambda$ and a set of keywords $q.\psi$. The objects returned are ranked according to a ranking function $f(\mathsf{Dist}(q,o), \mathsf{Sim}(q,o))$, where $\mathsf{Dist}(q,o)$ measures the Euclidian distance between $q.\lambda$ and $o.\lambda$ and $\mathsf{Sim}(Q,o)$ computes the text relevance between $q.\psi$ and $o.\psi$ using a language model.

A wide range of ranking functions can be adopted in SWORS, namely all functions that are monotone with respect to the distance function $\mathsf{Dist}(q,o)$ and the text relevance function $\mathsf{Sim}(q,o)$. Currently, SWORS computes a ranking score of an object o with regard to a query q as follows:

$$RS(q,o) = \alpha \frac{\mathsf{Dist}(q,o)}{maxD} + (1-\alpha)(1 - \frac{\mathsf{Sim}(Q,o)}{maxP}),$$

where $\alpha \in [0,1]$ is a parameter used to balance spatial proximity and text relevancy; where the Euclidean distance between q and o is normalized by $maxD$, which is the maximal distance between any two objects in D; where $maxP$ is used to normalize the probability score and is computed using the maximal language model of each word.

2.2 Spatial Group Keyword Query

Consider a spatial group keyword query $q = \langle q.\lambda, q.\psi \rangle$, where $q.\lambda$ is a location and $q.\psi$ represents a set of keywords. The spatial group keyword (SKG) query finds a group of objects χ, $\chi \subseteq D$, such that $\cup_{r \in \chi} r.\psi \supseteq q.\psi$ and such that the cost $\mathsf{Cost}(\chi)$ is minimized. Given a set of objects χ, the cost function has two components:

$$\mathsf{Cost}(q,\chi) = C_1(q,\chi) + C_2(\chi),$$

where $C_1(\cdot)$ is dependent on the distances of the objects in χ to the query object and $C_2(\cdot)$ characterizes the inter-object distances between the objects in χ.

This type of cost function is capable of expressing that result objects should be near the query location ($C_1(\cdot)$), that the result objects should be near to each other ($C_2(\cdot)$). We consider two instantiations of the cost function $\mathsf{Cost}(q,\chi)$. Consequently, we obtain two types of SKG queries.

TYPE1 SKG Query: The cost function of this type of query is defined as:

$$\mathsf{Cost}(q,\chi) = \sum_{r \in \chi} (\mathsf{Dist}(r,q))$$

The cost function is the sum of the distance between each object in χ and the query location. The TYPE1 SKG query may serve applications that involve travel or transportation from the query result locations to the query location, such as travel to a meeting point or the delivery of goods (e.g., take-out food).

TYPE2 SKG Query: The cost function of this type of query is defined as:

$$\mathsf{Cost}(q,\chi) = \max_{r \in \chi} (\mathsf{Dist}(r,q)) + \max_{r_1, r_2 \in \chi} (\mathsf{Dist}(r_1, r_2))$$

The first part of this function is the maximum distance between any object in χ and the query location q, and the second part is the maximum distance between two objects in χ (this can be understood as the diameter of the result). When there are multiple optimal groups of objects, we choose one such group at random. The TYPE2 SKG query may fit with applications where tourists plan visits to several points of interest.

3. SWORS PROTOTYPE

We cover the architecture and then the browser/client and server sides.

3.1 Architecture of SWORS

SWORS adopts the browser-server model for desktop and laptop computers, and it adopts the client-server model for mobile devices. The architecture is shown in Figure 1. The system offers browser-based interfaces for desktop and laptop users and a client for mobile users (using Android-based smartphones). Users input their queries through the web browser or the Android client application, and the queries are then sent to the server for processing. After the queries are processed, the results are sent back and displayed using Google Maps in the users' browser or client. On the server side, the spatial web objects are indexed by the IR-tree [2]. Specifically, queries are processed according to the user requirements (searching for single objects or a group of objects) utilizing a disk-resident IR-tree. Queries are sent from the browser or the client to the server by the HTTP post operation.

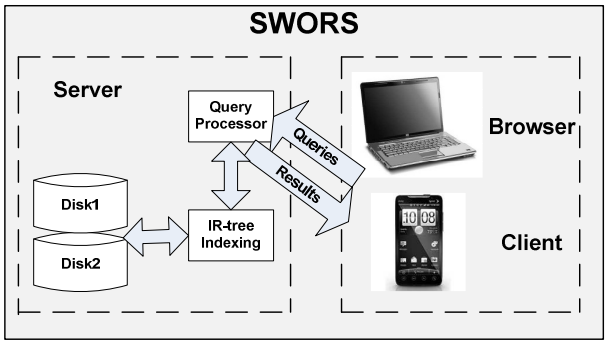

Figure 1: SWORS Architecture

3.2 Browser/Client Side

The browser side in computers and the client side in mobile devices provide interfaces to users for generating queries and viewing the returned spatial web objects. This component presents a map and provides interactions with the map using the Google Maps API.

Users specify a location and input a set of keywords that describe their queries through the web-based interfaces or the mobile client, where the query specifications are gathered. Users can provide their location information in two ways. First, they can click a location on Google Maps. The Google Maps API is then called to get the latitude and longitude of that location. Second, they can input the name of a location. Then a web service provided by Google that geocodes the location name is called to retrieve the latitude and longitude of the location. On the mobile platform, if users do not specify a location explicitly, their current location is detected by the mobile device and is used in the queries.

Queries are sent to the server by the HTTP post operation. The top relevant objects are retrieved by the server, and are displayed using Google Maps in the browser or client. Users can click the relevant objects shown on the map for more detailed information.

3.3 Server Side

The SWORS web server is built using JSP and Apache Tomcat. Once a query is received by the JSP server, the query processor implemented in Java is called to find the query result.

3.3.1 Data Storage

We organize the spatial web objects and process the queries utilizing the IR-tree, which is essentially an R-tree extended with inverted files. Each leaf node in the IR-tree contains entries of the form $(o, o.\lambda, o.di)$, where o refers to an object in dataset D, $o.\lambda$ is the bounding rectangle of o, and $o.di$ is an identifier of the description of o. Each leaf node also contains a pointer to an inverted file with the keywords of the objects stored in the node.

An inverted file index has two main components: (1) A vocabulary of all distinct words appearing in the description of an object. (2) A set of posting lists, each of which relates to a word t. Each posting list is a sequence of pairs $\langle o.di, w_{o,t}\rangle$, where $o.di$ is an identifier of an object o whose description contains the word t and $w_{o,t}$ is the weight of term t in the description of object o.

Each non-leaf node R in the IR-tree contains a number of entries of the form $(cp, rect, cp.di)$, where cp is the address of a child node of R, $rect$ is the minimum bounding rectangle (MBR) of all rectangles in entries of the child node, and $cp.di$ is an identifier of a pseudo text description that is the union of all text descriptions in the entries of the child node. The weight of each word t in the pseudo document referenced by $cp.di$ is the maximum weight of the word in the documents contained in the subtree rooted at node cp. Additional details are available elsewhere [1, 2].

The detailed descriptions of objects are stored in a MySQL database. After the result objects are found, we use their identifiers for obtaining their corresponding text descriptions from the database, and we return the information to the user.

3.3.2 Query Processing

LkT and SKG queries are processed utilizing the IR-tree. Since the problem of answering both types of SKG queries is NP-complete (the proof can be found in the literature [1]), we devise both exact and approximation algorithms for them in SWORS.

Queries are sent to the server by the HTTP post operation. Other user requirements are also attached with the query: 1) query type selection: the LkT query (searching for single objects) or the SKG query (searching for a group of objects); 2) algorithm type selection (for SKG queries): exact algorithms (asking for accurate results with longer query time) or approximation algorithms (asking for approximate results with fast response). Next, we briefly describe the algorithms for processing the queries in SWORS.

LkT Query Processing: The LkT query is processed using best-first traversal in the IR-tree. A priority queue is used to keep track of the IR-tree nodes and objects that have yet to be visited. When deciding which node to visit next, the algorithm picks the node that most likely provides the top-k objects among the set of all nodes that have yet to be visited. The algorithm terminates when k highest ranked objects (ranked according to the ranking function $RS(q, o)$) have been found.

SKG Query Processing: For the SKG TYPE1 query, an approximation algorithm with a provable performance bound and an exact algorithm based on dynamic programming are supported by SWORS.

The approximation algorithm follows the idea of solving the NP-hard weighted set cover problem. It selects the next best object greedily in each step and finally forms a group. The IR-tree is used to prune the search space.

The exact algorithm is devised based on the assumption that in some applications, the number of keywords in a query may not be large. It exploits dynamic programming and the IR-tree. The exact algorithm avoids enumerating the combinations of data objects in the database. Rather, it enumerates the query keywords and exploits a series of pruning strategies to reduce the search space. We utilize the IR-tree to retrieve only the objects that contain some query keywords. Objects are processed in ascending order of their distances to the query. When it is assured that no more better results exist, the reading of objects stops, and the best group is returned

SWORS supports a 2-approximation ratio algorithm based on the IR-tree for the SKG TYPE2 query. The algorithm first finds the nearest object for each keyword in the query.

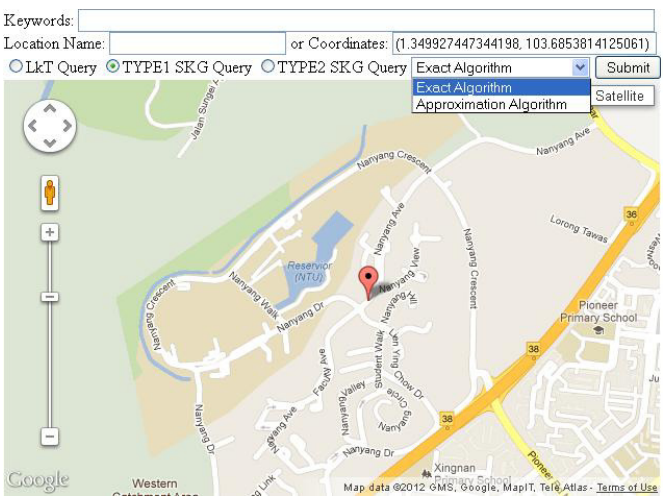

Figure 2: Desktop/Laptop Interface

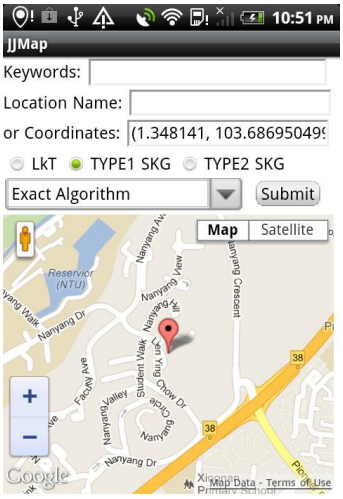

Figure 3: Android Smartphone Client

Next, the keyword only covered by the furthest object in this group is found. Each object containing such a keyword is checked in ascending order of its distance to the query. On each of them, a new query is formed and issued with the location of that object and the keywords of the original query. The nearest object for each keyword in the query to this object is then retrieved, and the cost of this group with regard to the original query is computed. Finally, the group with the best cost is returned.

We also support an exact algorithm that exploits the IR-tree and branch-and-bound search to prune the search space. This algorithm is based on the assumption that the number of keywords in a query is small. It first utilizes the approximation algorithm to derive an upper bound cost for the best group. A best-first search on the IR-tree is performed to enumerate the possible groups. The upper bound, which is updated when a better group is found, is used to prune the search space.

Presentation of the complete details of the algorithms can be found elsewhere [1, 2].

4. DEMONSTRATION AND DETAILS

The demonstration of SWORS is the first demonstration of a non-proprietary, open platform for the efficient retrieval of spatial web objects. Participants will be able to experience how the system can be used to retrieve objects according to various types of user requirements.

The interfaces of SWORS are developed for desktop/laptop computers and Android-based smartphones as shown in Figures 2 and 3, respectively. Users specify the keywords in the "Keywords" text box. They can provide the location information by either entering the name of a location in the "Location Name" text box or clicking a location on Google Maps (the latitude and longitude of the location as obtained using the Google Maps API is then shown in the "Coordinates" text box). On the mobile devices, if no location is specified explicitly, the user's current location is detected automatically. Users also need to choose the query type (LkT query, TYPE1 SKG query or TYPE2 SKG query). For the LkT query, the top returned relevant objects are listed in ascending order according to their ranking scores. If the user wishes to issue an SKG query, the user should select a corresponding algorithm (exact algorithm or approximation algorithm) for processing the query. The returned group of objects is shown on the map. Users can click one location in the map to see detailed descriptions.

We use three real-world data sets for the demonstration. The first dataset is crawled from Foursquare [1] in the region of Singapore. It consists of 47,653 points of interest, each of which contains a latitude and longitude, its categories (restaurant, mall, etc.), and a description provided by users who checked into the point of interest. The second one is a large dataset collected from Flickr [2] in the region of new York City in the United States, which contains over 1.5 million photos. Each photo is associated with a set of user-annotated tags and the latitude and the longitude of the place where the photo was taken. The third dataset contains about 179,000 points of interests with categories, names, and descriptions in Europe downloaded from PocketGPSWorld [3]. The demonstration will show that SWORS is able to process the spatial keyword queries effectively and efficiently over all three data sets.

5. REFERENCES

[1] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD*, pages 373–384, 2011.

[2] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.

[3] I. De Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.

[4] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. L. Lee, and X. Wang. IR-Tree: An efficient index for geographic document search. *TKDE*, 23(4):585–599, 2011.

[5] B. Yao, F. Li, M. Hadjieleftheriou, and K. Hou. Approximate string search in spatial databases. In *ICDE*, pages 545–556, 2010.

[1] `http://foursquare.com`
[2] `http://www.flickr.com`
[3] `http://www.PocketGPSWorld.com`