SOPS: A System for Efficient Processing of Spatial-Keyword Publish/Subscribe

Lisi Chen[1] Yan Cui[1] Gao Cong[1] Xin Cao[1]

[1]School of Computer Engineering, Nanyang Technological University, Singapore

lchen012@e.ntu.edu.sg, ycui3@e.ntu.edu.sg, gaocong@ntu.edu.sg, xcao1@e.ntu.edu.sg,

ABSTRACT

Massive amount of data that are geo-tagged and associated with text information are being generated at an unprecedented scale. These geo-textual data cover a wide range of topics. Users are interested in receiving up-to-date geo-textual objects (e.g., geo-tagged Tweets) such that their locations meet users' need and their texts are interesting to users. For example, a user may want to be updated with tweets near her home on the topic "dengue fever headache."

In this demonstration, we present SOPS, the Spatial-Keyword Publish/Subscribe System, that is capable of efficiently processing spatial keyword continuous queries. SOPS supports two types of queries: (1) Boolean Range Continuous (BRC) query that can be used to subscribe the geo-textual objects satisfying a boolean keyword expression and falling in a specified spatial region; (2) Temporal Spatial-Keyword Top-k Continuous (TaSK) query that continuously maintains up-to-date top-k most relevant results over a stream of geo-textual objects. SOPS enables users to formulate their queries and view the real-time results over a stream of geo-textual objects by browser-based user interfaces. On the server side, we propose solutions to efficiently processing a large number of BRC queries (tens of millions) and TaSK queries over a stream of geo-textual objects.

1. INTRODUCTION

Massive amount of data that are geo-tagged and associated with text information are being generated at an unprecedented scale. First, increasing volume of user generated content on the Web is being associated with geo-locations. Example user generated content includes geo-tagged micro-blogs (e.g., Twitter), photos with both tags and geo-locations in social photo sharing websites (e.g., Flickr), and check-in information on places in location-based social networks (e.g., FourSquare). Second, points of interests (POIs), such as shops and restaurant, are increasingly associated with text descriptions (e.g., reviews) in local search services.

These data featured with both textual content and geo-spatial content are referred to as geo-textual objects, and they can be modeled as geo-textual data streams. We consider the following real-world applications on such data streams. First, since social updates (e.g., Tweets) often offer the quickest first-hand reports of news events [6], a POI service provider (e.g., Yelp) may want to annotate each POI with its up-to-date relevant tweets in terms of both text relevance and spatial proximity. Second, users are interested in receiving up-to-date geo-textual objects such that their locations meet the spatial proximity requirement and their texts are interesting to the users [3]. In these applications, the number of POIs (resp. user queries) can be millions, and geo-textual objects may arrive in very high volume of tens of millions a day.

The number of POIs or continuous queries issued by users can be very large, which posts challenges for efficiently matching POIs or continuous queries over geo-textual objects. To develop an efficient solution to these applications, we propose a publish/subscribe system, where POIs or users' queries are subscriptions and geo-textual tweets are published items. Specifically, we consider two types of spatial-keyword publish/subscribe queries.

The first type of query is Boolean Range Continuous (BRC) Query [1]. The BRC query consists of two components, a boolean keyword expression and a spatial region. The answer to a BRC query comprises such geo-textual objects that satisfy the boolean keyword expression and fall in the specified spatial region.

Example 1: Consider a user who is interested in the real-time comments about *Philipp Lahm* in the 2014 FIFA World Cup semi-final between Brazil and Germany posted by the spectators on the spot when the match is in progress. She would like to receive all the tweets that contain relevant keywords and whose locations fall in a specific region. The following query might be posed: "Receive all tweets that contain *Lahm*, and are posted within 1km of the *Estadio Mineirao Stadium.* " □

However, sometimes a user may receive very few matching geo-textual objects or they may be overwhelmed by a huge volume of matching geo-textual objects. For the same reasons that search engines rank-order documents matching a query rather than employ the boolean retrieval model (e.g., the resulting number of matching documents of a boolean filter can far exceed the number a human user could possibly sift through [5]), we may want to rank-order geo-textual objects matching a query.

Example 2: Consider a user who is interested in the up-to-date tweets about the searching progress for the Malaysia Airline plane MH370 near the center of the planned search area. The following query might be submitted: "continuously feed me with new tweets whose ranking scores in terms of the distance to the query location $44°57'S$, $90°13'E$, the text relevance with the query keywords *MH370 plane rescue*, and the freshness are in the *top*-20. " □

In Example 2, three aspects are taken into account for evaluating the relevance with a geo-textual object: (1) text relevance;

Proceedings of the VLDB Endowment, Vol. 7, No. 13
Copyright 2014 VLDB Endowment 2150-8097/14/08.

(2) spatial proximity; and (3) recency. We define such query as Temporal Spatial-Keyword Top-k Continuous (TaSK) query that continuously maintains the up-to-date top-k results over a stream of geo-textual objects.

In order to process such queries efficiently, we build the SOPS system that employs the Inverted File Quad-tree (IQ-tree) [1], which is a dynamic index for managing BRC and TaSK queries. The SOPS system targets the efficient processing of both BRC and TaSK queries over a stream of geo-textual objects in real time.

This demonstration enables participants to view the real-time annotated geo-textual objects for each POI and formulate their queries to view the real-time results over a stream of geo-textual objects using Google Maps by browser-based user interfaces. On the server side, we organize the queries using the IQ-tree and match the geo-textual objects utilizing this index. Queries are sent from browsers to the server by the standard HTTP post operation.

The rest of the demonstration proposal is organized as follows. Section 2 introduce the formal definitions of geo-textual object, BRC query, and TaSK query. Section 3 presents the prototype of SOPS. Finally, Section 4 illustrates the demonstration details.

2. DATA MODEL AND QUERIES

We introduce the geo-textual object and define the BRC query and the TaSK query.

Definition 1: Geo-Textual Object. A geo-textual object is represented with a triple $o = \langle \psi, \rho, t_c \rangle$, where $o.\psi$ is a set of keywords, $o.\rho$ is a location point with latitude and longitude, and $o.t_c$ is the creation time of object o. □

In this demonstration, we consider a stream of geo-textual object data. For example, it can be geo-tagged tweets in Twitter, geo-tagged photos with tags in Flickr, check-ins with text descriptions in Foursquare, geo-tagged webpages, etc.

Definition 2: Boolean Range Continuous (BRC) Query. A BRC query is defined as a tuple $q = \langle \psi, r \rangle$, where $q.\psi$ is a set of query keywords connected by AND or OR semantics, and $q.r$ represents a spatial region. □

A user can submit a BRC query to the system, and then she continuously receives geo-textual objects in a timely fashion such that the retrieved geo-textual objects satisfy the boolean keyword expression $q.\psi$ and are located in the query range $q.r$.

Definition 3: Temporal Spatial-Keyword Top-k Subscription (TaSK) Query. A TaSK query $q = \langle \psi, \rho, k, \alpha \rangle$, where ψ is a set of query keywords, ρ is the query location, k is the number of results to be maintained, and $\alpha \in [0, 1]$ is a preference parameter that balances the importance between distance proximity and text relevance, aims to continuously feed the user with new geo-textual objects whose temporal spatial-keyword scores are ranked in the top-k. The temporal spatial-keyword score of a geo-textual object o at time t_e is defined as follow.

$$S_{tsk}(q, o, t_e) = S_{sk}(q, o) \cdot S_t(o.t_c, t_e), \quad (1)$$

where $S_{sk}(q, o)$ computes the spatial-keyword relevance between query q and object o and $S_t(o.t_c, t_e)$ computes the object recency. The spatial-keyword relevance $S_{sk}(q, o)$ between q and o is computed as follow.

$$S_{sk}(q, o) = \alpha \cdot S_p(dist(q.\rho, o.\rho)) + (1 - \alpha) \cdot TRel(q.\psi, o.\psi), \quad (2)$$

where $S_p(dist(q.\rho, o.\rho))$ is the *spatial proximity score* of the distance between query q and object o, and $TRel(q.\psi, o.\psi)$ indicates the *text relevance* between q and o. □

Intuitively, given a stream of geo-textual objects, a TaSK query is to continuously retrieve k objects over time such that these objects are relevant to the query keywords, their locations are close to the query location, and they are fresh.

The spatial proximity score is calculated by the normalized Euclidian distance: $S_p(dist(q.\rho, o.\rho)) = 1 - dist(q.\rho, o.\rho)/dist_{max}$, where $dist(q.\rho, o.\rho)$ is the Euclidian distance between q and o, and $dist_{max}$ can be the maximal possible distance in the spatial area. The text relevance can be computed using any information retrieval model. We use language models in this work.

The recency of object o is calculated by the following exponential decay function

$$S_t(o.t_c, t_e) = D^{-(t_e - o.t_c)} \quad (3)$$

where D is base number that determines the rate of the recency decay. The function is monotonically decreasing with $t_e - o.t_c$. It is introduced in [4] and is applied (e.g., [6]) as the measurement of recency for stream data.

Property: Our scoring method is general and guarantees that the relative ranking of two different objects w.r.t. a query is consistent as time passes, i.e., if $S_{tsk}(q, o_j, t) > S_{tsk}(q, o_k, t)$, then $\forall \Delta t > 0$ we have $S_{tsk}(q, o_j, t + \Delta t) > S_{tsk}(q, o_k, t + \Delta t)$.

With this property, we eliminate the need of re-ranking query results over time. However, the ranking scores of objects in results will decrease with time, and new geo-textual objects may have larger ranking scores and become one of results.

3. SOPS PROTOTYPE

We cover the framework and then the browser and server sides.

3.1 Framework of SOPS

SOPS adopts the browser-server model. The architecture is shown in Figure 1. Users submit their queries through the web browser, and the queries are then sent to the server and inserted into the IQ-tree for continuously receiving the results over a stream of geo-textual objects. For each new geo-textual object, the system traverses the IQ-tree and finds the queries that have the object as a result. Each result object is sent back to the user who issues the query and is displayed on Google Maps in the users' browser.

In our SOPS prototype, the processing schemes of the BRC query and TaSK query are different. For the BRC query, it is inserted into the IQ-tree and then the geo-textual objects satisfying the spatial and text boolean query expressions are returned as results. As for the TaSK query, its processing is more complicated, its top-k results are firstly initialized by traversing the object index as illustrated in Figure 1. The object index stores the geo-textual objects not older than a specified time period. Then, the query is inserted into the IQ-tree. When a new object arrives, it is stored into the object index component at first. Then, the IQ-tree is utilized to find the TaSK queries whose top-k results include the new object, and their top-k results are updated and sent to the users who issue these queries.

3.2 Browser Side

The browser side provides interfaces to users for submitting queries and viewing the returned objects. This component provides interactions with the map through the Google Maps API.

When submitting a BRC query, the user specifies a location, a radius, and a set of keywords connected by AND or OR that describes filtering requirements. Users may also input their email addresses for receiving the query results. As for a TaSK query, the user specifies a location, a set of keywords, the result size ("k")

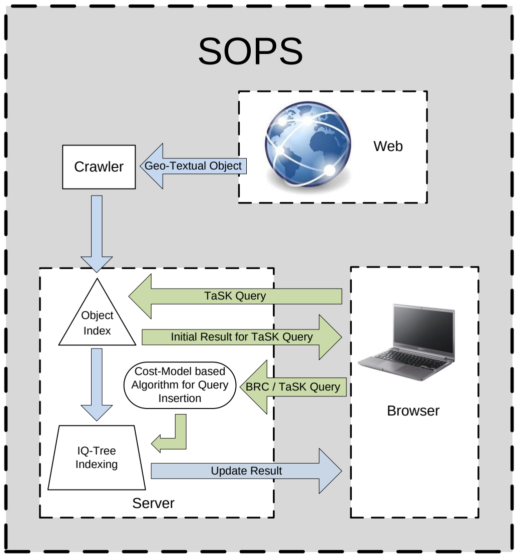


Figure 1: SOPS Architecture

that indicates the number of geo-textual objects maintained, and a preference parameter that balances the importance between spatial proximity and text relevance.

Queries are sent to the server by the HTTP post operation. After the query is indexed by the server, each new geo-textual object that is a result is displayed using Google Maps in the browser.

3.3 Server Side

3.3.1 Overview

The web server of SOPS is built using JSP and Apache Tomcat. Once a query is received by the JSP server, it will be inserted into the IQ-tree through a cost-model based algorithm. When a new geo-textual object is received by the server, it will be inserted into the object index and then the query processing algorithm implemented in Java is invoked to find the queries that have the new object as a result.

In our applications, the typical arrival rate of geo-textual objects (e.g., tweets) is in the scale of millions a day, while new TaSK queries are added at the rate of tens of thousands a day, and we may serve millions of BRC and TaSK queries at one time. We thus focus on a scalable solution to match and maintain the up-to-date results for a large number of queries over a data stream of geo-textual objects. Millions of queries can easily fit into the available memory of modern servers. Hence, our system is developed under this setting.

3.3.2 Indexing and Query Processing

We use a structure that combines the quad-tree and inverted file to organize the geo-textual objects in memory, and we utilize the IQ-tree to organize the BRC and TaSK queries in memory, and match the queries with geo-textual objects. Note that the IQ-tree [1] can also be disk-based. The IQ-tree is essentially a quad-tree, which recursively divides the rectangular spatial area into four mutually-exclusive congruent rectangles, each node of which is enriched with reference to an inverted file for the queries that are associated at the node.

Note that each BRC query or TaSK query, may be associated with multiple non-overlapping Quad-cell nodes. The association of queries and the nodes of the IQ-tree is a challenging issue, which affects the system performance significantly. We propose techniques for associating BRC and TaSK queries with the nodes of the IQ-tree based on the cost models following our previous work [1,2].

An inverted file comprises a vocabulary of terms, and a set of postings lists, each associated with a term. Each posting records the information of a query that contains a particular term under a particular Quad-cell node. We use two optimizations of inverted file for indexing BRC query and TaSK query respectively, which are stated as follows.

For indexing BRC queries, we apply the *Ranked key method* to organize the query keywords to improve the matching processing. Specifically, if the query keywords are connected by AND semantics, we store the query into the postings list of which word is the least frequent among all the query keywords; and the other query keywords are stored in the posting of the query. If the query keywords are connected by OR semantics, we index the query using standard technique of the inverted file, i.e., the query will appear in the postings list of each keyword in the query. Since a BRC query q can be associated with multiple non-overlapping Quad-cell nodes, any set of non-overlapping nodes on different levels of the IQ-tree that intersect the query region of q could be associated with q. So we need to find an association scheme with the minimum expected cost for each query. The cost is incurred by (1) index update: inserting and deleting a BRC query in the IQ-tree, and (2) object matching: traversing the inverted lists of the IQ-tree for matching the BRC queries over incoming geo-textual objects. We utilize a cost model based algorithm [1] that finds a set of nodes for associating each query by balancing the trade-off between the two types of costs.

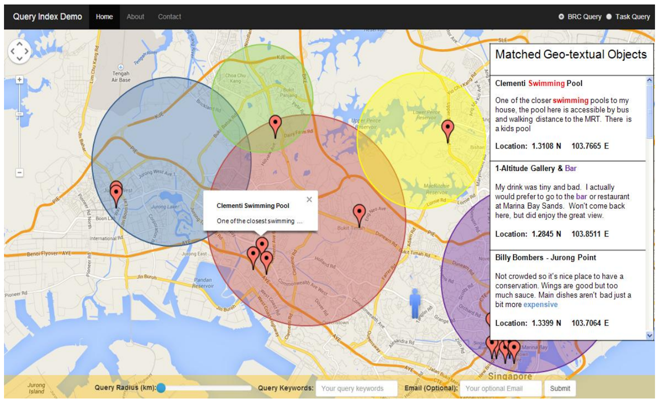
For indexing TaSK queries, our high-level idea comprises three steps: (1) For each TaSK query q we choose a set of Quad-cell nodes, which may be from different levels of the IQ-tree and can cover the whole spatial area, and generate the postings of q for each selected node; (2) Given q, for each Quad-cell node n we compute the minimum value of text relevance such that if the relevance between q and a new object o that falls in the region of n is greater than the minimum value, o will be used to update the current top-k results of q; (3) We group the postings in each postings list into blocks, each of which contains a specified number of postings. The purpose of using the block based inverted file is to skip blocks in a postings list such that an incoming object cannot be a result for all the queries in the block based on the lower bound of the score of the kth result of each query in the block.

For processing BRC queries, when a new geo-textual object o arrives, we need to find the Quad-cell nodes of the IQ-tree whose regions cover the location of o. Then for each of such nodes, for each keyword w in o, we retrieve its postings list. Each posting in the postings list corresponds to a BRC query q, and we check if the spatial region of query q covers o and the keyword expression of q is matched by o. As for processing TaSK queries, when a new geo-textual object o arrives we traverse the block based inverted file under the Quad-cell nodes of the IQ-tree whose regions cover o. Specifically, at each level of Quad-cell node containing o we traverse its postings lists of all the words contained in object o simultaneously based on the Document-at-a-Time (DAAT) technique with forward skipping [5].

The details of the indexing structure and algorithms for processing BRC and TaSK queries can be found elsewhere [1, 2].

4. DEMONSTRATION AND DETAILS

In this demonstration, participants will be able to experience how the system can be used for issuing continuous queries to retrieve geo-textual objects over the data stream. The browser interfaces



Matched Geo-textual Objects

Clementi Swimming Pool

One of the closer swimming pools to my house, the pool here is accessible by bus and walking distance to the MRT. There is a kids pool

Location: 1.3108 N 103.7665 E

1-Altitude Gallery & Bar

My drink was tiny and bad. I actually would prefer to go to the bar or restaurant at Marina Bay Sands. Won't come back here, but did enjoy the great view.

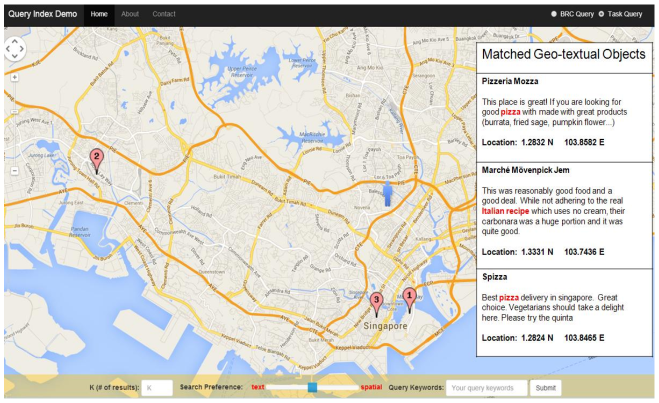
Location: 1.2845 N 103.8511 E

Billy Bombers - Jurong Point

Not crowded so it's nice place to have a conservation. Wings are good but too much sauce. Main dishes aren't bad just a bit more expensive

Location: 1.3399 N 103.7064 E

Figure 2: Browser Interface for BRC Query



Matched Geo-textual Objects

Pizzeria Mozza

This place is great! If you are looking for good pizza with made with great products (burrata, fried sage, pumpkin flower...)

Location: 1.2832 N 103.8582 E

Marché Mövenpick Jem

This was reasonably good food and a good deal. While not adhering to the real Italian recipe which uses no cream, their carbonara was a huge portion and it was quite good.

Location: 1.3331 N 103.7436 E

Spizza

Best pizza delivery in singapore. Great choice. Vegetarians should take a delight here. Please try the quinta

Location: 1.2824 N 103.8465 E

Figure 3: Browser Interface for TaSK Query

of SOPS developed for answering BRC query and TaSK query are shown in Figures 2 and 3, respectively. Users may choose the query type in the navigation bar.

Submitting a BRC query: Users specify the query location by clicking a location on Google Maps (the latitude and longitude of the location is obtained using the Google Maps API). Users then need to specify the radius of the query region using the slider bar. The colored circles in Figure 2 indicate the regions of BRC queries issued through the browser. To specify the query keywords, users can input the keywords in the "Query Keywords" text box and connect them by either AND or OR ("Space" will be regarded as AND semantics by default). If users hope to receive email notifications when each qualified geo-textual object arrives, they may indicate their email addresses in the "Email" text box. When a BRC query is submitted, the browser will display the location and text of qualified geo-textual objects in a real-time manner.

Submitting a TaSK query: For submitting a TaSK query, users specify the query location and the query keywords. They also need to specify the number of results to be maintained for this query in the "K" text box. To specify the preference parameter, users can drag the corresponding slider bar that balances the weight between text relevance and spatial proximity. If the slider is dragged to "text", the ranking function just consider the score of text relevance, and the score of spatial proximity between the query and the geo-textual object will not be considered any more. If the slider is dragged to "spatial", only the score of spatial proximity will be considered in the ranking function while processing this TaSK query. The up-to-date top-k results for each query are maintained.

Datasets and Demonstrations: We use four real-world data sets for the demonstration. The first dataset is the businesses and reviews of Yelp[1] in Singapore, which contains 11,240 businesses and 20,764 reviews. Each business, which can be considered as a Point of Interest (POI), contains the business name, categories, and the location on Google Map. Each review has a location, text description and the timestamp of creation. The second dataset contains about 179,000 POIs with locations, names, categories, and descriptions in Europe downloaded from PocketGPSWorld[2]. The third dataset contains about 120,000 POIs in Foursquare[3] with locations, names, and categories in the U.S.A. The fourth dataset contains 10,000,000 geo-tagged tweets crawled from Twitter[4] with both location and text. Reviews on Yelp and geo-tagged tweets on Twitter are regarded as geo-textual data streams.

We demonstrate two scenarios. First, participants can submit their own BRC and TaSK queries through the browser to continuously receive qualified reviews or tweets. Second, we annotate each POI with relevant geo-tagged tweets in real time. This is attained by regarding each POI as a BRC query or a TaSK query. Participants may zoom in on the map and select the POI they are interested in to see the annotated tweets.

To demonstrate that SOPS offers scalability and is capable of good performance for handling a large number of queries over a high volume of geo-textual data stream, we initialize the IQ-tree with 10,000,000 randomly generated queries (including both BRC queries and TaSK queries) and then we vary both the average arrival rate of tweets and the number of queries. Based on our evaluation result, SOPS is able to efficiently process 20 million subscribe queries over a stream of more than 50 million geo-textual objects per day.

5. ACKNOWLEDGEMENTS

This work is supported in part by a grant awarded by a Singapore MOE AcRF Tier 2 Grant (ARC30/12) and a Singapore MOE AcRF Tier 1 Grant (RG66/12).

6. REFERENCES

[1] L. Chen, G. Cong, and X. Cao. An efficient query indexing mechanism for filtering geo-textual data. In *SIGMOD Conference*, pages 749–760, 2013.

[2] L. Chen, G. Cong, and X. Cao. Temporal spatial-keyword top-k publish/subsribe, Technical Report, School of Computer Engineering, Nanyang Technological University. 2014.

[3] G. Li, Y. Wang, T. Wang, and J. Feng. Location-aware publish/subscribe. In *KDD*, pages 802–810, 2013.

[4] X. Li and W. B. Croft. Time-based language models. In *CIKM*, pages 469–475. ACM, 2003.

[5] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[6] A. Shraer, M. Gurevich, M. Fontoura, and V. Josifovski. Top-k publish-subscribe for social annotation of news. *PVLDB*, 6(6):385–396, 2013.

[1]`http://www.yelp.com`

[2]`http://www.PocketGPSWorld.com`

[3]`http://foursquare.com`

[4]`http://twitter.com`