A System for Region Search and Exploration

Kaiyu Feng[1,2] Kaiqi Zhao[2] Yiding Liu[2] Gao Cong[2]
[1]LILY, Interdisciplinary Graduate School. Nanyang Technological University, Singapore
[2]School of Computer Engineering, Nanyang Technological University, Singapore
{kfeng002@e., kzhao002@e., liuy0130@e., gaocong@}ntu.edu.sg

ABSTRACT

With the increasing popularity of mobile devices and location based services, massive amount of geo-textual data (e.g., geo-tagged tweets) is being generated everyday. Compared with traditional spatial data, the textual dimension of geo-textual data greatly enriches the data. Meanwhile, the spatial dimension of geo-textual data also adds a semantically rich new aspect to textual data. The large volume, together with its rich semantics, calls for the need for data exploration. First, it has many applications to retrieve a region for exploration that satisfies user-specified conditions (e.g., the size and shape of the region) while maximizing some other conditions (e.g., the relevance to the query keywords of the objects in the region). Second, it is useful to mine and explore the topics of the geo-textual data within a (specified or retrieved) region and perhaps a timespan. This demonstration proposal presents the main ideas of our system, the RegIon Search and Exploration System (RISE), for efficiently supporting region search and exploration, and our demonstration plan.

1. INTRODUCTION

Due to the prominence of mobile devices and increasing popularity of location-based services (*e.g., Foursquare* (`www.foursquare.com`), *Yelp* (`www.yelp.com`)), massive amount of data that contains both textual and geographical information is being generated at an unprecedented scale, such as geo-tagged tweets, and *points of interest* (POIs) associated with category information and textual descriptions. We refer to such data as geo-textual data.

Geo-textual data from various sources is often characterized by big volume. For example, around 10 million geo-tagged tweets are generated every day in Twitter[1] and 7 million check-ins were submitted on October 3rd, 2015[2]. The availability of such large-scale geo-textual data calls for the need for *region search* and *region exploration*.

[1]https://www.mapbox.com/blog/twitter-map-every-tweet/
[2]http://blog.foursquare.com/post/130625318273/7-million-check-ins

Proceedings of the VLDB Endowment, Vol. 9, No. 13

1.1 Region Search

There are many applications to retrieve a region for exploration that satisfies user-specified conditions while maximizing some other conditions. We use an example to explain the problem.

Example 1: George is planning a trip to Singapore and he is very interested in Singapore culture. Rather than visiting a region in Singapore with a specific attraction, he wishes to visit a region that has the most diverse collection of services and attractions related to "Singapore" (*e.g.,* Singapore food, Singapore art show, Singapore history museum, and Singapore music bar). This will enable him to experience many different attractions and services in one place without the need to travel to different regions to experience them all together. How can George select the "most diversified region" in Singapore? □

We refer to the problem in Example 1 as the *Best Region Search* (BRS) query. Specifically, consider a set of POIs and a set of geo-tagged tweets. Given a set of keywords, a query rectangle of size $a \times b$, *BRS* aims to find a rectangular region of size $a \times b$ such that the diversity of the region is maximized.

In the *BRS* problem, we can find infinite number of different rectangular regions of the given size in the space and it is prohibitively expensive to consider all of them. Furthermore, a user may search for the best region in an exploratory manner. That is, she may initiate a search with a specific query rectangle as input, view the corresponding results, iteratively refine the query rectangle (by increasing or decreasing a or b), and execute the refined search until she is satisfied with the search results. Such an exploratory framework demands techniques that can efficiently process the *BRS* queries over large volumes of geo-textual objects(e.g., POIs).

1.2 Region Exploration

Given a specified or retrieved region, users may want to mine different properties, e.g., topics [10] or frequent words [7], of the geo-textual data, within the region and perhaps a timespan to help them explore the region.

Example 2: George wants to visit Clarke Quay area in Singapore, where there are restaurants, bars, dance clubs, and shopping malls. He wants to know what are the hot topics (e.g., dining or movie) in this area during a specified period. □

We refer to the problem in Example 2 as *Topic Exploration within a Region and a Timespan* (TE) query. Given a region, an integer k, and a time period $[t_b, t_e]$, *TE* aims to mine topics within the specified region and timespan. The list of topics is complementary to the result of *BRS* query. It helps users better understand the region they found.

In *TE* problem, it is challenging to mine topics efficiently. A straightforward method works as follows: Given a user specified

region and timespan, we train a topic model on the geo-textual data falling in the region and the timespan using the existing techniques, such as Latent Dirichlet Allocation (LDA) [2]. However, mining topics from geo-tagged data is very time-consuming. A user may also mine topics within the region in an exploratory manner. That is, she may mine the topics within the region in a specific time period, view the topics, and iteratively refine the query (by changing the time period or the region) and execute the *TE* query until she is satisfied with the mined topics. Thus such an exploratory framework also demands techniques that can efficiently process the *TE* queries over large volumes of geo-textual objects.

1.3 Contributions

We develop the RegIon Search and Exploration System (RISE)[3], which can be used to handle *BRS* queries and *TE* queries.

To process the *BRS* queries, we first develop an exact algorithm. In the exact algorithm, we propose several novel concepts, based on which effective pruning techniques are proposed to reduce the search space. The exact algorithm is still slow on large dataset, and to further improve on efficiency, we develop a constant-bounded approximate algorithm. Specifically, we select a set of spatial points from the space which together preserve some properties of the geo-textual objects instead of considering all the objects. Then the exact algorithm is invoked on the selected spatial points. The approximate algorithm is very efficient. It can find an answer in less than 1 second on a real-life dataset from Meetup containing 589,715 geo-textual objects.

To efficiently mine the topics within a user specified region or the region returned by a BRS query, we develop a novel and efficient sampling algorithm with a bounded error to combine two LDA topic models learnt from two document sets, both falling in the specified region and time period. We also develop a new approach to partitioning the collection of the geo-tweets into an Octree for indexing the tweets. The proposed sampling algorithm is at least an order of magnitude faster than the online-LDA [1], the state-of-the-art algorithm that samples topics for incrementally maintaining LDA models.

2. RELATED WORK AND NOVELTY

Work Related to Region Search The existing systems [4] support querying for a region satisfying user-specified conditions. However, they are built on a modern array DBMS, which is not optimized for spatial objects. In addition, their system does not support submodular aggregate functions.

Work Related to Region Exploration The existing work on analyzing and exploring multi-dimensional text database is closest to our problem [5, 6, 8, 9]. However, some of them [5, 6, 9] do not consider the semantic meaning (i.e., topics) of each term in documents, and some of them [8] require the topic model to be generated from all documents in the datasets. Our system is different in that, it mines topics from the documents falling in a given region and timespan. Because topics within a region could be more specific than topics learnt from the whole set, our system discovers topics specific to the given region. To the best of our knowledge, no existing work studies the problem of mining region-specific topics in an exploratory manner.

3. BRS QUERY AND TE QUERY

We first introduce the geo-textual object and our Best Region Search query.

[3]The system is available at `http://spatialkeyword.sce.ntu.edu.sg/rise`

DEFINITION 1. **Geo-Textual Object** *A geo-textual object is a pair $o = \langle \phi, \rho \rangle$, where $o.\phi$ is a set of keywords, and $o.\rho$ is a location point with latitude and longitude.*

In this demonstration, we consider two types of geo-textual object: POIs and geo-tagged tweets. Apart from the set of keywords ϕ and the location point ρ, POIs and geo-tagged tweets are enriched with a set of category attributes and a creation time, respectively. Specifically, each POI is a triple $p = \langle \phi, \rho, A \rangle$, where A is a set of category attributes, e.g., "restaurant", and each geo-tagged tweet is a triple $tweet = \langle \phi, \rho, t_c \rangle$, where t_c is the time when the tweet was created.

We next define the best region search query.

DEFINITION 2. **Best Region Search (BRS) Query** *Consider a set of spatial objects O and a submodular monotone aggregate function $f : 2^{|O|} \rightarrow \mathbb{R}$. Given a* BRS *query $q = \langle a \times b, T \rangle$, where $a \times b$ is the size of the rectangular region that users want to find, and T is a set of keywords that the users are interested in, we aim to find a rectangular region r of size $a \times b$ in the space, such that*

$$r = \arg\max_{r} f(O_{r,T}),$$

where $O_{r,T}$ is the set of spatial objects which are inside the region r and contain keywords in T.

We can use the number of total check-ins in the region, referred to as **popularity**, as the aggregate function to select the most popular region. We can also use the number of different category attributes of the POIs in the region, referred to as **diversity**, as the aggregate function to select the most diversified region (as shown in Example 1). Both of them are important properties that should be considered. Thus, we combine both **popularity** and **diversity** and define a new aggregate function, named **PD** score, as follows.

DEFINITION 3. **PD Score** *Consider a set of POIs O_i. Let A_{O_i} be the set of category attributes associated with the POIs in O_i. Let $C(o)$ be the number of check-ins associated with a POI o. For an attribute a, we define its popularity $P(a) = \max_{o \in O_i(a)} C(o)$, where $O_i(a)$ is the set of POIs in O_i that have attribute a. The PD score of a set of objects O_i is defined as:*

$$S(O_i) = \sum_{a \in A_{O_i}} P(a).$$

The PD score is a submodular monotone function. In this demonstration, we use region score as the aggregate function. Other submodular monotone aggregate functions like **popularity** and **diversity** can also be easily adopted into our system. We omit the detailed definition and properties of BRS problem which could be found elsewhere [3].

DEFINITION 4. **Topic Exploration within a Region and a Timespan (TE) Query** *Let R be a rectangular region, and $[t_b, t_e]$ be a time period.* TE *query is to mine k topics from geo-tagged tweets whose locations fall in R and creation time falls in $[t_b, t_e]$, based on a given topic model.*

4. RISE PROTOTYPE

4.1 Framework of RISE

Figure 1 shows the system architecture of RISE. The *Query Processor* module handles the *BRS* queries, while the *Topic Miner* module handles the *TE* queries. RISE adopts the browser-server model. A user can submit his query through the web browser. Then

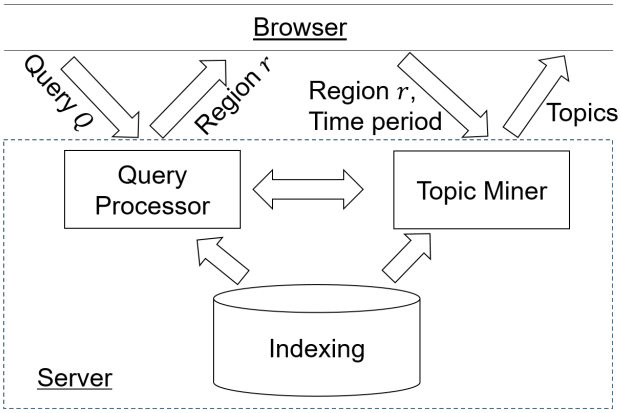

Figure 1: RISE Architecture

the query is sent to the *Query Processor* module. The *Query Processor* module accesses the *Indexing* module and finds a rectangular region according to users' requirements. The *Query Processor* module then passes the region to the *Topic Miner*. By accessing the *Indexing* module, the *Topic Miner* module extracts the topics in the region. Then the region together with the popular topics is returned to users. In addition, users can submit different time intervals and regions to the *Topic Miner* module to extract popular topics.

4.2 The Indexing Module

The *Indexing* module comprises two components: *POI Indexing* and *Geo-Tweets Indexing*.

In *POI Indexing*, we use a Quadtree to index all POIs according to their location in the space in order to efficiently access POIs. Each node of the Quadtree is enriched with a reference to an inverted file for the objects contained in the sub-tree rooted at the node. An inverted file consists of the following two components: (1) A vocabulary for all distinct terms, and (2) a set of posting lists, each of which relates to a term t. Each posting list is a sequence of objects containing term t. In addition, for each node v in the Quadtree, we maintain a point, denoted by $v.p$. Specifically, if v is an internal node, let $v.p$ be the center point of the corresponding region. If v is a leaf node, let $v.p$ be the spatial object in the corresponding region.

In *Geo-Tweets Indexing*, we use an Octree to index the tweets because instant messages such as tweets are often sensitive to time. Octree is an extension of Quadtree in a three-dimensional space. To support efficient and effective online topic mining, we build an Octree with pre-trained topic models by considering both accuracy and space constraints. Specifically, we greedily partition an Octree cell into 8 subcells until the number of expected pre-trained models exceeds a space threshold. We determine the partition position by considering two objectives for better accuracy: 1) minimizing the word overlaps among the subcells; and 2) balancing the number of documents among the subcells. During the partition, we pre-train topic models for some cells of different levels in the Octree. This guarantees that the online training can use the pre-trained information to obtain an approximate topic model with a bounded error.

4.3 The Query Processor Module

Queries are sent from the browser to the server by the HTTP post operation. Then the processor finds a rectangular region of a given size according to users' requirements. Finally, the results are sent back to users and displayed on the map using Google Maps API.

The goal of RISE is to find the best region of a given size. Two algorithms are involved in our RISE prototype:

- **SliceBRS** algorithm: This algorithm finds the exact answer to the *BRS* problem.
- **CoverBRS** algorithm: Since in most cases, slight imprecision to the solution is acceptable, this algorithm finds an approximate answer to the *BRS* with performance guarantees.

SliceBRS algorithm In this algorithm, we propose several new concepts, including "maximal regions" and "maximal slabs". Based on these concepts, we cut the space into slices and prune unnecessary slices and maximal slabs to reduce the search space. It takes $O(n \times n_s)$ time to find an exact answer to the query, where n is the number of POIs that contains the query keywords, and n_s is the number of maximal slabs that we actually processed.

CoverBRS algorithm The key idea behind the approximate algorithm is as follows: Consider a set O of POIs, each of which contains the terms specified by users. Instead of taking all POIs in O into account, we only select a small set of spatial points from the space. The selected points together preserve some properties of O, such that the rectangular region found on the selected points is an approximation of the result on O with a constant performance guarantee. This idea has two potential benefits: (1) The size of the selected points is smaller; (2) The selected points are sparser than those in O. Thus, the exact algorithm runs much faster on the selected POIs. These benefits enable us to answer the *BRS* query more efficiently.

The *CoverBRS* algorithm comprises three steps: (1) select a set of spatial points that can preserve some properties, (2) generate a new instance of the *BRS* problem according to user's query and selected points, and (3) invoke the exact algorithm to solve the new instance.

Specifically, in the first step, given the Quadtree maintained by the *Indexing* module, we first compute the level l of the Quadtree that we should access. Then we collect all centers of regions in all internal nodes at level l and the POIs in all leaf nodes at any level not lower than l. Note that we only consider POIs which contain all query keywords and all other POIs are ignored in this step. For each selected spatial point p, we use p to represent a set of POIs $\{o_1, ..., o_j\}$.

In the second step, we need to define the category attributes for each selected point based on which POIs can be represented by this point. We also revise the size of the query rectangle to get a new instance of *BRS* query. The exact results of the new instance is an approximate answer to the original *BRS* query bounded by a constant 4.

The complexity of *CoverBRS* is $O(n + n^t \times n_s^t)$, where n is the number of POIs that containing all query keywords, n^t is the number of selected spatial points and n_s^t is the number of maximal slabs that we actually processed. The *CoverBRS* algorithm can find a $1/4$-approximate answer to the *BRS* query.

4.4 The Topic Miner Module

This module supports mining topics from a user specified region. The user can also constrain the topic mining tasks by specifying a time interval. The topic mining method returns the top-k (e.g., 100) topics estimated from the documents that lie in the user specified region and time interval. Since training a topic model for each topic mining query has efficiency issue in exploratory topic mining tasks, we design a combining method [10] that leverages pre-trained topic models in the Octree index (Section 4.2), to significantly reduce the training time. The combining algorithm for two cells in the online topic learning phase uses one pre-trained topic model as prior knowledge to resample the topics for the other one. The sampling process follows the idea that the pre-trained models have already

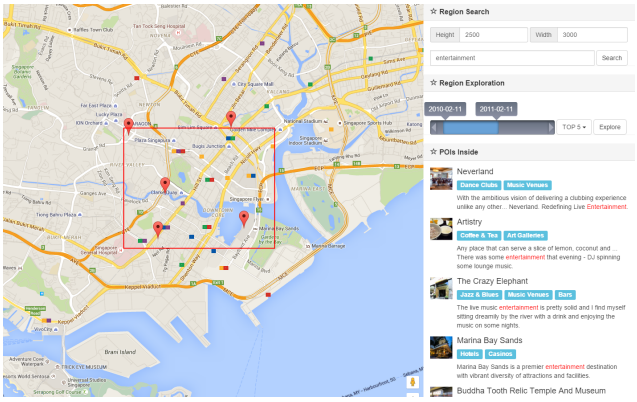

Figure 2: Searching best region

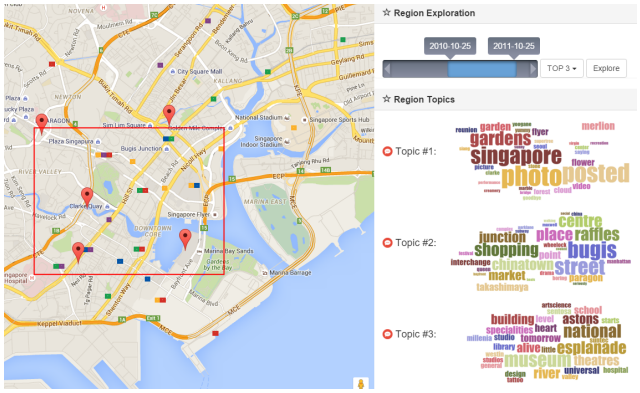

Figure 3: Mining popular topics

captured the correlation between words in the form of topics and assigned each word token a topic label. We make use of this information to group the word tokens by topic and sample topics for each word token group instead of sampling individual word tokens. While the complexity of training a LDA model on two document sets is $O((|W_1| + |W_2|)KI)$, our sampling algorithm reduces the complexity to $O(K(K + |M_2| + |V_2|)I)$, where W_1 and W_2 are sets of word tokens in the two document sets, M_2 is the smaller document set, and V_2 is its vocabulary. The size of M_2 and V_2 are much smaller than W_1 and W_2. As a result, our algorithm is at least an order of magnitude faster than the one that samples topics for individual word tokens. We prove that the approximation error of our sampling algorithm is proportional to the number of word tokens that appear in both cells [10], and thus we optimize the partition in Section 4.2 by considering the overlap of word tokens of the resulting subcells.

The details of the indexing structure, algorithms for searching best region and algorithms for mining the top-k popular topics in the region can be found elsewhere [3, 10].

4.5 Browser Module

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

When generating a query, users enter the width and height of the rectangular region that they want to find, and a set of keywords that describes their interests. The query is then sent to the server for processing.

After the query is processed by the server, a region of the given size is returned and displayed on the map. Users can click the region and specify a time period to get a topic summary of the region, which describes what people in the region were talking about in the specified time period.

5. DEMONSTRATION

Datasets We use two real-life datasets for POIs with textual description: Yelp and Meetup. For Meetup, we collect 589,715 venues in total. Each venue is associated with 14.7 keywords on average. For Yelp, we collect 48,753 venues in total. Each venue is associated with 48 keywords on average. We use one real-life dataset, Twitter, for answering the *TE* query. We collect 4 million geo-tagged tweets posted in Singapore from 144k users.

Region Search In this scenario, user can search for the best region. First of all, users should specify the height and width of a query rectangle, and possibly a set of keywords to submit a region search query. Note that by specifying a set of keywords, the user can include the categories that they are interested in into the diversity factor. Then the *Query Processor* module will run either the exact or the approximate algorithm to search for the region and return it to the user. Finally, the region will be displayed on the map using a red rectangle. The POIs containing the query keywords will be displayed at the right panel of the browser (see Figure 2). Users can revise the region search query and issue a new query to compare the results.

Region Exploration In this scenario, users can explore hot topics in the region. Firstly, the user can use the retrieved region from region search, or he can draw a new rectangle on the map as a query region. Secondly, the user can specify the time period by moving the slider bar. Then he can select how many topics he would like to see. Our system will extract the topics within the region and time period. For each topic, a word cloud will be generated to represent its semantic meaning. The word clouds are displayed at the right panel of the browser. Figure 3 shows three example topics, i.e., sightseeing, shopping, and art, mined from the query region and time interval. For each topic, we can also display a set of representative tweets for the topic. Users can also specify two regions and compare the topics of the two regions.

Acknowledgments This work is supported in part by a Tier-1 grant awarded by Ministry of Education Singapore (RG22/15).

6. REFERENCES

[1] L. AlSumait, D. Barbar, and C. Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *ICDM*, pages 3–12, 2008.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[3] K. Feng, G. Cong, S. S. Bhowmick, W.-C. Peng, and C. Miao. Towards best region search for data exploration. SIGMOD. ACM, 2016.

[4] A. Kalinin, U. Cetintemel, and S. Zdonik. Searchlight: enabling integrated search and exploration over large multidimensional data. *Proceedings of the VLDB Endowment*, 8(10):1094–1105, 2015.

[5] C. X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao. Text cube: Computing ir measures for multidimensional text database analysis. In *ICDM*, pages 905–910. IEEE, 2008.

[6] A. Simitsis, A. Baid, Y. Sismanis, and B. Reinwald. Multidimensional content exploration. *Proceedings of the VLDB Endowment*, 1(1):660–671, 2008.

[7] A. Skovsgaard, D. Sidlauskas, and C. S. Jensen. Scalable top-k spatio-temporal term querying. In *ICDE*, pages 148–159. IEEE, 2014.

[8] D. Zhang, C. Zhai, and J. Han. Topic cube: Topic modeling for olap on multidimensional text databases. In *SDM*, volume 9, pages 1124–1135. SIAM, 2009.

[9] B. Zhao, X. Lin, B. Ding, and J. Han. Texplorer: keyword-based object search and exploration in multidimensional text databases. In *CIKM*, pages 1709–1718. ACM, 2011.

[10] K. Zhao, L. Chen, and G. Cong. Topic exploration in spatio-temporal document collections. SIGMOD. ACM, 2016.