Fuzzy Keyword Search on Spatial Data

Sattam Alsubaiee, Chen Li

Department of Computer Science, University of California, Irvine, CA 92697, USA
{salsubai, chenli}@ics.uci.edu

Abstract. In recent years, many websites have started providing keyword-search services on maps. In these systems, users may experience difficulties finding the entities they are looking for if they do not know their exact spelling, such as the name of a restaurant. In this paper, we present a solution to support fuzzy keyword search on spatial data. We combine a spatial index structure with inverted indexes on grams to efficiently answer fuzzy queries on maps. We show two system prototypes to demonstrate the practicality of our solution.

1 Motivation

Many websites based on geographical information nowadays support keyword search on their data such as business listings and photos. Such services accept queries consisting of two parts: a set of keywords and a spatial location. The goal is to find objects with these keywords close to the location. Such a query is called a *spatial-keyword* (SK) query [1]. There are several local-search websites, such as Google Maps, Yahoo! Local, Bing Maps, Yellow Pages, and MapQuest Maps. At such a website, a user might look for a restaurant called "Aomatsu" close to Irvine in California. The website returns business listings close to the city that match the keywords. Another example website is the service by Flickr that supports location-based photo search (http://www.flickr.com/map). A user may ask for photos about the "Coliseum Stadium" close to Los Angeles.

Users often do not know the exact spelling of keywords. For example, the user may mistype a query as (`aumatso restaurant`) *near* (`Irvine`, `CA`) when looking for the restaurant `Aomatsu`. Similarly, a user could mistype the word "`coliseum`" and submit a query: (`colisum stadium`) *near* (`Los Angeles`, `CA`). It is important to find relevant answers to such mistyped queries. Unfortunately, most existing location-based systems do not provide correct answers to a query even with only a single typo. Table 1 shows how several systems behaved for five mistyped variations of the query "`aomatsu restaurant`" as of June 20, 2009. In most cases, the search engines either returned an empty answer or gave irrelevant results. Both Google and Yahoo could suggest alternative queries, but they very often could not give the right suggestion. We also experimented with the Flickr Maps photo search engine and saw similar limitations. An interesting observation is that during the development of our work, the results of these systems kept changing. For instance, the evaluation results as of September 2009 had more "no-results" cases.

Table 1. Results of Local-Search Engines for Mistyped Queries (as of June 20, 2009)

Search Engine	**Results of Mistyped Queries**				
	`aumatso restaurant`	`aomatso restaurant`	`aumatsp restaurant`	`amatsu restaurant`	`aumatso`
Yahoo! Local	●	✓	●	●	●
Bing Maps	●	●	●	●	●
Yellow Pages	●	●	●	✘	●
MapQuest Maps	✘	✘	✘	✘	●

● : No results ✓ : Correct suggestion ✘ : Wrong suggestion/answer

In this paper, we study how to solve this problem by supporting fuzzy keyword search on spatial data. Given a query with keywords and a location, we want to find objects close to the location with those keywords, even if those keywords do not match exactly. Thus we can find relevant objects for the user even in the presence of typos in the query or data. Notice our approach is more powerful than the approach of suggesting an alternative query (the "Did you mean" feature used by many systems). The latter can only suggest a new query, while our approach can find relevant answers, even if the answers' keywords are only similar to those of the query.

2 Problem Formulation and Our Solution

Formulation: Consider a collection of spatial objects $o_1, \ldots, o_n$, and each object has a textual description (a set of keywords) T_i and a location L_i. A query consists of the following: $Q = \langle Q_s, Q_t \rangle$, where Q_s is a spatial region such as a rectangle or a circle. Q_t is a fuzzy-keyword condition, which consists of a set of keywords and an edit-distance threshold δ. Our goal is to find the objects in the collection such that each of them r is within the region Q_s. In addition, for each keyword k in Q_t, the object r has a keyword d in its description, such that the edit distance between k and r is within the threshold δ. For simplicity, we assume the threshold is a constant, and our results can be easily generalized to the case where the threshold varies based on the length of the keywords. A related problem is fuzzy string search: given a collection of strings, how to efficiently find those that are similar to a given query string? Many algorithms have been proposed to answer fuzzy keyword queries using inverted lists of grams [2]. Several algorithms have been proposed in the literature to answer spatial-keyword queries by assuming exact matching of keywords [1, 3]. A recent paper [4] also studies how to support fuzzy keyword search on spatial data. Their approach is probabilistic, and does not guarantee to find *all* the answers to a query.

Our solution: We use an R*-tree to index the objects based on their spatial attribute. Our solution extends naturally to other tree-based structures, such as kd-trees and quadtrees. Each node in the tree stores the keywords of the spatial objects in its leaf nodes. To support fuzzy keyword search, we choose nodes in

the tree to build gram-based inverted indexes for their stored keywords. In this paper, we choose one level of the tree and construct gram indexes for all the nodes at that level, denoted by L.

We answer a fuzzy spatial-keyword query as follows. Let Q be a query with a spatial condition Q_s and a fuzzy-keyword condition Q_t. Intuitively, the algorithm traverses the tree top-down. Before reaching level L, where the gram-based inverted indexes reside, the algorithm only relies on the spatial information of each node to decide which nodes to traverse. The rationale is that higher levels can have many keywords, and it is computationally expensive to do pruning based on the condition Q_t by finding similar keywords. At level L, for each candidate node, the algorithm uses the node's gram inverted index to find keywords that satisfy the fuzzy-keyword condition Q_t, i.e., finding keywords that are similar to at least one keyword in Q_t according to the edit-distance threshold. This set of similar keywords, denoted by C, is propagated in the later process of the traversal in order to prune branches in the tree.

We studied how to choose the level L of tree nodes to construct gram indexes. Notice that at each tree node, its stored keywords is the *union* of the keywords of its leaf-node objects. If multiple objects have the same keyword, this keyword is stored only once in the common ancestors of their leaf nodes. In particular, the root of the tree ($L = 1$) has all the keywords in the dataset. We can see a trade-off between the query performance and the size of the gram inverted indexes. As L increases, the total number of keywords on which we need to build gram inverted indexes increases. Thus the total size of the gram inverted indexes will increase. Meanwhile, the performance of finding similar keywords from a gram inverted index is very related to the size of the index.

3 Demonstration Description

We used two real datasets to develop two prototypes for demonstration. The first dataset was a multimedia metadata collection extracted from Flickr pages, called "CoPhIR Test Collection" (http://cophir.isti.cnr.it). We processed the dataset to extract the photos taken in the U.S. based on their latitude and longitude values. Moreover, we used the keywords in the title, description, and tags of a photo as its textual attribute. The final dataset had about two million objects, with a size of 300MB. Each record had a URL corresponding to the photo or a page including the photo. The second dataset had geographical objects (such as lakes and hills) obtained from http://www.geonames.org/. We used the objects residing in the U.S., and the final dataset had about 1.8 million objects. The total data size was 90MB. In the experiments, we built inverted indexes using 2-grams.

Both systems provide an interface similar to existing local-search and photo-search services on maps. Each interface has a map and two input boxes, one for textual keywords and one for a location. The map is using the Google Maps API, and can display the search results for a query. We also use the Google Maps Geocoder API to obtain the latitude and longitude of the entered location. Once

the user clicks the search button, the objects satisfying the query conditions will be shown as red markers on the map. If the user clicks a marker, the information about the corresponding object (e.g., a photo) will be displayed. Some markers overlap with each other, and we grouped these near-by markers under a green marker. If the user clicks a green marker, the map will zoom in to show the included objects. We will use the prototypes to demonstrate the capabilities and advantages of supporting fuzzy keyword queries. Fig. 1 shows a screenshot of the results page for a query with a mistyped keyword on the first dataset.

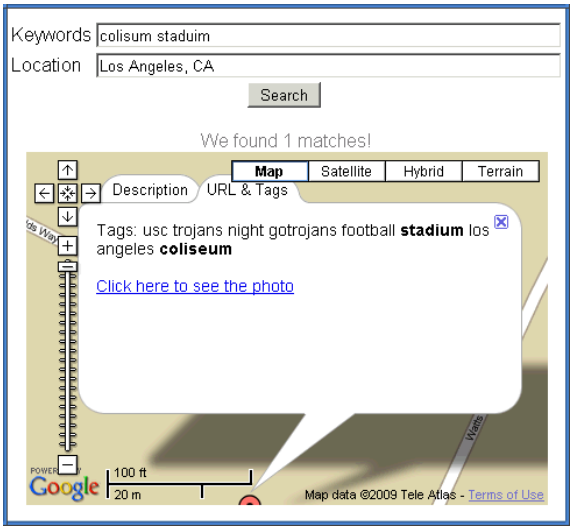

Fig. 1. A screenshot of our system on the CoPhIR dataset for answering the mistyped query "`colisum stadium` *near* `Los Angeles, CA`".

Acknowledgements: We thank Kensuke Ohta for his discussions in this work.

References

1. Hariharan, R., Hore, B., Li, C., Mehrotra, S.: Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In: SSDBM, p. 16. (2007)
2. Li, C., Lu, J., Lu, Y.: Efficient merging and filtering algorithms for approximate string searches. In ICDE, pp. 257–266. (2008)
3. Felipe, I. D., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In ICDE, pp. 656–665. (2008)
4. Yao, B., Li, F., Hadjieleftheriou, M., Hou, K.: Approximate string search in spatial databases. In ICDE. (2010)