SpSJoin: Parallel Spatial Similarity Joins (Demo Paper)*

Jaime Ballesteros
jball008@cis.fiu.edu

Ariel Cary
acary001@cis.fiu.edu

Naphtali Rishe
rishen@cis.fiu.edu

School of Computing and Information Sciences
Florida International University
Miami, FL 33199

ABSTRACT

A spatial similarity join of two geospatial datasets finds pairs of records that are simultaneously similar on spatial and textual attributes. Such join is useful for a variety of applications, like data cleansing, record linkage, duplications detection and geocoding enhancement. Efficient techniques exist for the individual joins on either spatial or textual attributes. However, the combined problem has received much less research attention. This paper presents the *SpSJoin* (Spatial Similarity join) system to fill in this need. *SpSJoin* is a platform that merges geospatial and text processing techniques for efficiently performing spatial similarity joins. The platform leverages parallel computing with MapReduce to tackle scalability issues in joining large datasets. The efficiency of the proposed techniques are experimentally validated with a join case for improving the geolocation of entities in a real geospatial dataset with referential entities of another dataset.

1. INTRODUCTION

In modern geographical databases, records contain textual and spatial attributes to describe characteristics and location of real-world entities. When the location of the records has low precision, e.g. geolocated at the center of the city, their location may be enhanced by finding their *most similar* records in another database, known to have high location precision. For instance, Figure 1 shows sample records of *Physicians* database, geolocated at city center level precision and *Yellow Pages* database with high geolocation precision. Intuitively, the most similar object of physician "*John F. Smith MD*" is "*John Smith MD*" in Yellow Pages, since both names are very similar and geographically closer. Therefore, finding the most similar pairs between two geographical databases requires a composite join operation that considers both types of attributes, textual and spatial.

Such type of join, namely *Spatial Similarity join*, has received much less attention in the research community than individual joins on either textual or spatial attributes. In the textual case, the degree of resemblance in a *similarity join* [1] [7] is measured by a similarity function, e.g. Jaccard coefficient or Levenshtein distance, and pairs that satisfy a user-defined similarity threshold are included in the output. Recently, parallel processing with MapReduce, a parallel programming model proposed by Google [3], has been explored to tackle the scalability problem of this type of joins [6]. In the spatial case, a *spatial join* [4] between two geographical datasets matches records based on their spatial attributes. The spatial relation may be expressed in several ways, e.g. distance threshold or polygon overlap.

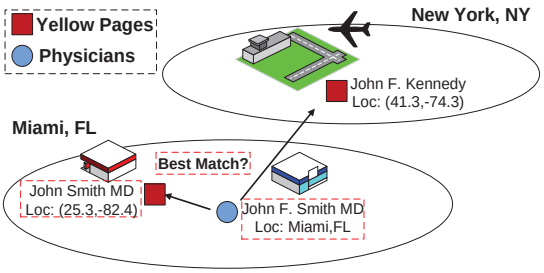

Figure 1: Best match for physician "*John F. Smith*".

Direct application of either spatial join or similarity join techniques to solve the spatial similarity join problem has the disadvantage of potentially generating lots of pairs that do not satisfy the composite constraint; for example, in Figure 1 several similar physician names and yellow page contact persons may be located far away from each other, e.g. "*John F. Kennedy*" in New York, but we are interested only in the geographically nearest pair. Also, when a threshold is predefined for either similarity or spatial joins, some records may not find their most similar pair when they do not satisfy the threshold. It is then up to the user to define an appropriate distance or similarity threshold even when there is no knowledge of the precision and quality of the data. In addition, as geolocation data is rapidly increasing in databases, scalability in processing spatial similarity joins is a top concern.

Spatial similarity joins have generally the same applications as similarity joins, including data cleansing and record linkage. In addition to geolocation enhancement, this join might be used in disaster management applications, e.g.

*This research was supported in part by NSF grants CNS-0821345, HRD-0833093, IIP-0829576, IIP-0931517, CNS-1057661, IIS-1052625, CNS-0959985, CNS-1126619. Ariel Cary was supported by an FIU dissertation year fellowship

ACM SIGSPATIAL GIS '11, November 1-4, 2011. Chicago, IL, USA

joining 911 call records with Nationwide cadastre and White Pages databases to pinpoint massive emergency events.

Contributions: We propose a combined similarity-based approach to solve the Spatial Similarity Join problem. We developed an algorithm that leverages the MapReduce parallel programming model [3] to handle large amounts of geographical data, tackling the scalability problem. We implemented the SpSJoin system, a platform for performing and analyzing results of spatial similarity joins on large geographical datasets.

2. SYSTEM ARCHITECTURE

The SpSJoin system is divided into four components. Figure 2 shows the proposed architecture for our system. The *Data Repository* stores the geographical databases used by the system and supports data persistency required by the interacting modules. The *Spatial Similarity Join* module performs the join and returns the result set that is indexed by the *Query Processing* module. Finally, the *Data Visualization* module presents an interface to the user for displaying and analyzing the join results.

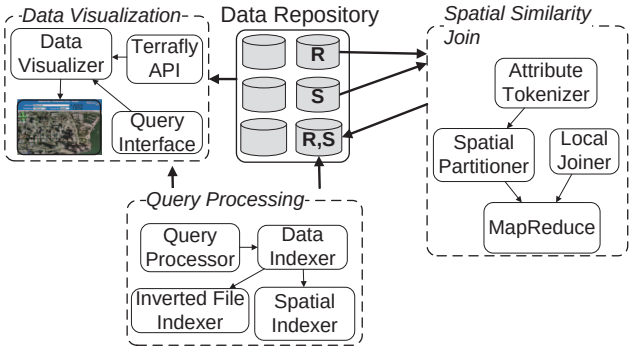

Figure 2: SpSJoin System Architecture

2.1 Data Repository

The data repository contains several geographic datasets used in different GIS applications. Data comes from different sources, including the Internet and public and private sources, that may or may not need additional geographic location processing. Examples of datasets found in the repository include *Hotels*, *Crime Data*,*Places and Landmarks*, etc., all of them containing different attributes and geographic location. Figure 3 shows the *Physicians* and *Yellow Pages* datasets with some of their attributes and spatial location.

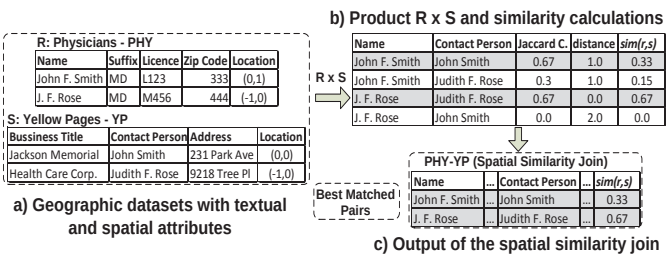

Figure 3: Example of a *Spatial Similarity Join*. Table PHY-YP contains the join result.

2.2 Spatial Similarity Join

Intuitively, a Spatial Similarity Join finds pairs of objects from two spatial datasets, a *target* and a *source*, in which every pair represents a match of an object in the target with the *most related* object in the source. Relatedness between objects is modeled with a composite similarity function that combines spatial and textual attributes. For instance, in Figure 3b, the similarity of a pair is calculated by combining the distance of the objects with their textual similarity on *Name* attribute in Physicians and *Contact Person* in Yellow Pages. The most related pairs from the Cartesian product are the ones with the highest value given by the similarity, $sim(r, s)$, function, e.g. $\langle$"*John F. Smith*", "*John Smith*"$\rangle$ and $\langle$"*J.F. Rose*", "*Judith F. Rose*"$\rangle$. Next, we present the problem statement and describe our approach for processing spatial similarity joins efficiently.

Notation. We denote our input datasets as R (*target*) and S (*source*). Without loss of generality, records in these datasets are tuples of the form $o = \langle a, p\rangle$, where a denotes a textual attribute and p is a point in the space that denotes the location of the object o. In practice, objects may contain additional textual attributes, which we omit to simplify the explanation. MBR refers to the Minimum Bounding Rectangle that encloses a set of objects. Given two objects r and s, we refer to the function $sim_t(a_r, a_s)$ as the textual similarity between attributes a_r and a_s, and $dist(p_r, p_s)$ as the distance between points p_r and p_s. We denote $sim(r, s)$ as the composite similarity function in the problem statement.

2.2.1 Problem Statement

Given two datasets R and S and a composite similarity function $sim(r, s) \in [0, 1]$, that combines spatial and textual similarity, the problem of *Spatial Similarity Join* finds the set of pairs $(r, s) \in R \times S$, such that $sim(r, s) = \max_{s' \in S}\{sim(r, s')\}$. We say that s is the *most related* object of r found in S and the pair (r, s) is a *best matched pair*.

Choosing an adequate $sim(r, s)$ function is challenging since each type of attribute has its own semantics and independent similarity values. Therefore, careful analysis of the datasets is required. For example, if R and S are known to have very precise spatial attributes, then $sim(r, s)$ may give less importance to the textual attributes.

2.2.2 Processing Spatial Similarity Joins

In our approach, $sim(r, s)$ meets the criterion that similarity of pairs of proximal objects must be higher than objects located far away from each other. We defined the following similarity function

$$sim(r, s) = \frac{sim_t(a_r, a_s)}{1 + dist(p_r, p_s)} \quad (1)$$

Where $sim_t(a_r, a_s)$ is a textual similarity function (we used the *Jaccard* coefficient in our experiments, $sim_t(a_r, a_s) = \frac{|a_r \cap a_s|}{|a_r \cup a_s|}$) and $dist(p_r, p_s)$ is a distance function (we used *Great Circle* distance since geographical objects are located with latitude and longitude). In general, if an object r has two possible matching objects s and s' with equal similarity value (i.e. $sim(r, s) = sim(r, s')$), the pair with minimum distance is considered the better pair.

When processing spatial similarity joins in large datasets, scalability is a key challenge. Our algorithm leverages parallel computing with MapReduce, which has proven its effec-

tiveness in large-scale data intensive problems [3].

The join process is divided into two main phases: a *Spatial Filtering* phase and an *Expansion* phase. In the Spatial Filtering phase, the entire set of records is partitioned w.r.t. their spatial attribute. The rationale is that geographically proximal object pairs are more likely to generate higher similarity values, using Equation 1. In this way, potential best matches are co-located in the same partition, filtering out pairs with low similarity value whose evaluation is not necessary, e.g. far away objects do not represent the same real world entity. Since each partition may contain some local best pairs that may have globally best matches, i.e. with increased similarity value, the Expansion phase gradually expands the search space of each partition using an upper bound *Expansion Region*. Object pairs are reprocessed iteratively on adjacent geographical regions until their similarity value cannot be improved anymore, i.e. the best pairs are found, or the expansion region covers all universe of objects. We illustrate the join execution with an example, shown in Figure 5, that describes the workflow of the process. We denote clusters of records as C_i, $i = \{1, 2, 3\}$, and sets L_i^j as local output in cluster C_i at iteration j. Final join output is denoted as L.

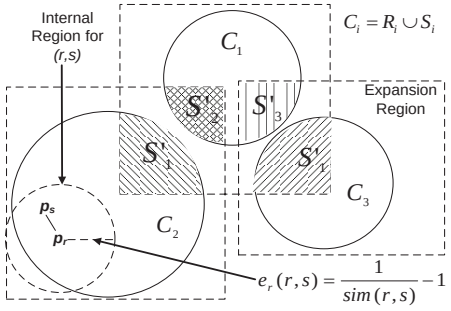

Figure 4: Dataset clustering. Clusters C_i are formed after Spatial Filtering phase. The dotted-line squares represent the MBRs of the upper bound expansion region for C_i

Spatial Filtering phase. Figure 5 part **(I)**. The *Spatial Partitioner* component is used for partitioning the entire set of records. It is expressed as a MapReduce job that clusters $R \cup S$ in parallel using a clustering algorithm; in our experiments, we used the X-means clustering technique [5]. Figure 4 shows the spatial layout of the three clusters in this example: C_1, C_2 and C_3. Note that each C_i is expressed in Figure 5 as $R_i \cup S_i$.

Expansion phase. Figure 5 part **(II)**. Each cluster C_i is processed locally in parallel using several expansion iterations. Each iteration of our example is described next.

Iteration 1. For each cluster C_i, the *Local Joiner* component joins R_i and S_i using a nested-loop approach; we implemented the Local Joiner using a modified version of the fuzzy join proposed in [6], leveraging the MapReduce framework. Mappers tokenize textual attributes from records in $R_i \cup S_i$ and generate record projections for each token, tagged with the relation name. Reducers receive records that share the same token, sorted by relation (S_i first), and records in R_i are combined with records in S_i. To accelerate the process, records in S_i are indexed using their spatial attibute. For every record in R_i, the spatial index filters records in S_i that

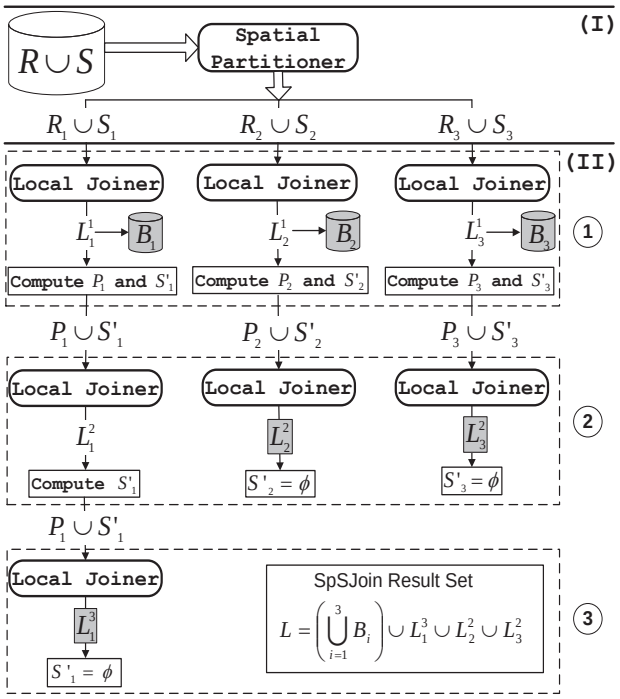

Figure 5: Example workflow for SpSJoin. Spatial Filtering phase is shown in step (I). The Expansion phase (II) performs iterations 1-3.

will not improve in the combined similarity. The combined similarity is computed for candidate pairs and the pair with the highest $sim(r, s)$ is kept. The output of the Local Joiner L_i^1 is the set of local best matched pairs found in cluster C_i.

In order to prepare for the next iteration, the input $P_i \cup S'_i$ needs to be calculated. We observed that each pair (r, s) in L_i^1 defines an *internal region*, as shown in Figure 4, with center p_r of r and radius e_r[1]. The union of all internal regions defines the upper bound *Expansion Region* for the cluster, in which objects from R_i may find better matches. Since the Expansion Region may overlap adjacent clusters, objects in pairs with internal regions that lie within the cluster's MBR will not find a better match and the corresponding pairs are stored in the B_i database as part of the final output. This reduces the size of the input in the next iteration. With the remaining pairs, objects in R_i are extracted and stored in P_i, which need further iterations. Finally, the system identifies the nearest cluster C_k, that overlaps the Expansion region, and stores the overlapping objects from S_k in S'_i. In Figure 4 for example, the nearest cluster of C_1 is C_3, so S'_1 is the set of records from S_3 in the shaded region of C_3.

Iteration 2. Each Local Joiner receive $C_i = P_i \cup S'_i$ as input and joins P_i and S'_i as in the previous iteration. Ouput pairs in L_i^2 that improved their similarity are updated as the new best pair matches. If further clusters need to be explored, the next nearest cluster that overlaps the Expansion region is identified and S'_i is computed as above. Else, the

[1]From Eq. 1, to improve a current $sim(r, s)$ with s_2, at best $sim_t(a_r, a_{s_2}) = 1$, hence $sim(r, s) \leq \frac{1}{1+dist(p_r, p_{s_2})}$. It follows that $dist(p_r, p_{s_2}) \leq \frac{1}{sim(r,s)} - 1 = e_r(r, s)$.

Table 1: Geographical databases of *Physicians* (PHY) and *Yellow Pages* (YP) used in experiments.

Database	Records	Joining Attributes	
		Textual	Spatial
PHY	2 millions	name	zip
YP	20 millions	contact person	location

local process finishes its execution. In Figure 4, Expansion regions for clusters C_2 and C_3 do not expand anymore so L_2^2 and L_3^2 are part of the final output. On the other hand, Expansion region of C_1 overlaps C_2 so P_1 requires further processing. S'_1 is now the set of records from S_2 in the shaded region of C_2

Iteration 3. Local Joiner is called again with the new input $C_i = P_i \cup S'_i$ and the output L_i^3 is generated. In our example, the Expansion region for cluster C_1 has no more overlapping clusters to cover, hence set L_1^3 is part of the final output. Since no clusters need further expansion, the process terminates and the join result set L is complete. Shaded blocks in Figure 5 form the final output of the join.

2.3 Query Processing

The *Query Processing* module , Figure 2, is used primarily by the *Data Visualizer* component to retrieve records of joined databases (generated by the *Spatial Similarity Join* module). This module executes spatial queries with non-spatial constraints posted by users for join quality inspection. Attributes in the join result are first indexed using a hybrid data structure that leverages R-trees and inverted files [2] by the *Data Indexer*. Second, the *Query Processor* parses a user query to identify the query window (geographical region) and (optionally) non-spatial constraints, and it uses the hybrid index structure to efficiently retrieve records. For instance, in the join example of Figure 3, joined records of physicians with last name "*Smith*" and located in "*Miami, FL*" are displayed in Figure 6.

2.4 Data Visualization

The *Data Visualization* module displays the results of spatial similarity joins on a map. Figure 6 shows the general user interface of the system. Aerial and satellite imagery as well as user interface widgets are provided by the TerraFly system[2] via its public API.

When the user selects a location to visualize, the currently displayed map portion determines the query window that will be submitted to the *Query Processing* module. Then, users pick a previously joined database from a database drop-down list to visualize its records. Optionally, users can include keywords in the query to locate specific objects for inspection. Records that match the query criteria are displayed as pairs, visually distinguished by circles and diamonds, united by lines and enclosed in rectangles. Users can click on individual object icons to display detailed information about the pair the object participates on.

3. DEMO DESCRIPTION

The setup for the demonstration is as follows. First, two real geographical databases, *Physicians* (target) and *Yellow Pages* (source), were joined with the SpSJoin operator. The database sizes and joining attributes are shown in Table 1. Objects in the databases represent real-world entities located in the United States. For example, YP entries include medical professionals of various specialities, which are expected to match with records in PHY. Jaccard coefficient and Great Circle distance were used to compute the similarity of textual and spatial attributes, respectively. The data was provided by the HPDRC laboratory[3]. Second, joined records were stored in a third database PHY-YP inside our *Data Repository*, and its attributes were indexed by the *Data Indexer* component. The use case demonstrates the improvement of geolocation precision in the *Physicians* database with matching objects in the *Yellow Pages* database; initially, records in PHY were geolocated to the center of their ZIP codes.

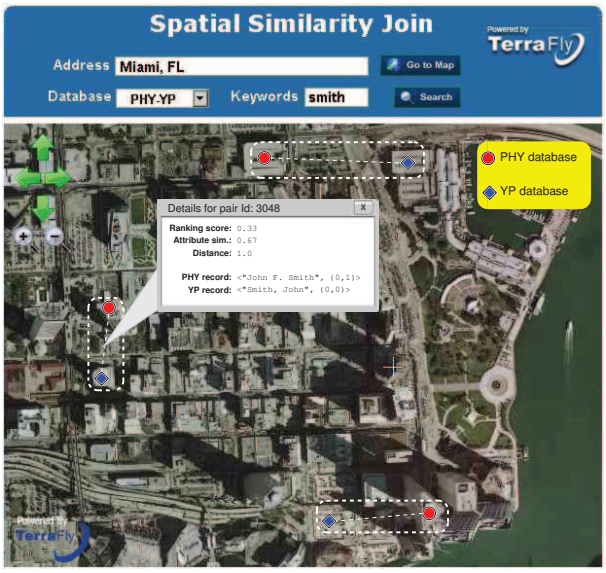

Figure 6: Data visualization of joined records.

During the demonstration, users will have the opportunity to interact with the system by visualizing the joined data in the PHY-YP database as shown in Figure 6.

4. REFERENCES

[1] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *VLDB'06*.

[2] A. Cary, O. Wolfson, and N. Rishe. Efficient and scalable method for processing top-k spatial boolean queries. In *SSDBM'10*, pages 87–95.

[3] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM 2008*, page 51(1).

[4] E. H. Jacox and H. Samet. Spatial join techniques. *ACM TODS'07*.

[5] D. Pelleg and A. W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *VLDB 2000*.

[6] R. Vernica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using mapreduce. In *SIGMOD'10*.

[7] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *WWW'08*.

[2]http://terrafly.fiu.edu

[3]http://hpdrc.fiu.edu