GeoDDupe: A Novel Interface for Interactive Entity Resolution in Geospatial Data

Hyunmo Kang[2], Vivek Sehgal[1], Lise Getoor[1, 2]
Computer Science Department[1]
And Institute for Advanced Computer Studies[2]
University of Maryland, College Park, MD 20742
{kang@cs.umd.edu, viveks@cs.umd.edu, getoor@cs.umd.edu}

Abstract

Due to the growing interest in geospatial data mining and analysis, data cleaning and integration in geospatial data is becoming an important issue. Geospatial entity resolution is the process of reconciling multiple location references to the same real world location within a single data source (deduplication) or across multiple data sources (integration). In this paper, we introduce an interactive tool called GeoDDupe which effectively combines automatic data mining algorithms for geospatial entity resolution with a novel network visualization supporting users' resolution analysis and decisions. We illustrate the GeoDDupe interface with an example geospatial dataset and show how users can efficiently and accurately resolve location entities. Finally, the case study with two real-world geospatial datasets demonstrates the potential of GeoDDupe.

1. Introduction

The recent rapid growth in the availability of geospatial data has resulted in significant interest in tools which support geospatial data mining and analysis tasks. In order to make these tools work effectively and provide accurate analytic information, the underlying data must be clean. Unfortunately, this is rarely the case. Like many other data sources, geospatial databases often contain multiple uncertain and imprecise references to the same real world locations and require the process of merging them (*deduplication*). Furthermore, the availability of geospatial data from various sources often requires the process of matching multiple references to actual location entities across datasets (*integration*).

Geospatial Entity Resolution is the process of determining a single consolidated collection of "*true*" locations from a collection of database sources referring to geospatial locations [28]. In other words, it is the process of reconciling location references to the same real-world location entity, which may exist within and across datasets. The geospatial entity resolution problem can be described more formally as follows:

There exist two geospatial datasets $\boldsymbol{A}$ and $\boldsymbol{B}$ developed by independent sources. Each dataset is a collection of location references, each of which corresponds to some real-world location entity. The goal of entity resolution is to find pairs of locations $\{l_i, l_j\}$, such that **($l_i, l_j \in A$ or $l_i, l_j \in B$)** (*deduplication*), or **(($l_i \in A$ and $l_j \in B$) or ($l_i \in B$ and $l_j \in A$))** (*integration*), where l_i and l_j correspond to the same real world location. Each location reference l_i is typically described by a set of features such as l_i = ***[location name, spatial coordinates (e.g. latitude and longitude), location type (e.g. dune, city, river, airport, etc.), demographic information (e.g. population), etc].***

Because of an increasing need for robust and accurate merging and matching techniques, there has been extensive work on entity resolution in various areas. However, previous work on entity resolution has primarily focused on non-spatial data attributes in medical data [13], census data [29], bibliographic data [15][23], and natural language [32][2]. In addition, traditional entity resolution approaches use similarity metrics which compare only the attributes of the references; while the entity's relationship context can also provide useful information to the resolution process [15][14][21]. Furthermore, most existing entity resolution methods use automated approaches which are not perfect and often face a precision-recall trade-off problem. That is, if the methods are tuned to have high precision, they leave many duplicates in the database. On the other hand, if they are tuned to have a high recall, they mistakenly merge nodes that are not duplicates.

In this paper, we take a different approach and designed a novel interface *GeoDDupe,* inspired by our prior interactive tool for entity resolution in social network data (*DDupe* [18]), for resolving location entities in geospatial data. *GeoDDupe* is designed to focus on both spatial (point-based geospatial) and non-spatial data while making use of the relational information of locations for geospatial entity resolution. *GeoDDupe* provides access to sophisticated and automated entity resolution algorithms based on the location attributes, which enables users to easily identify the potential duplicates from the geospatial datasets. In addition *GeoDDupe* provides a simple, stable, and meaningful network visualization, which displays the neighborhood context for potential duplicate locations, as well as a spatial network visualization in order to help users make their final resolution decisions.

2. Related Work

There has been extensive work on finding and cleaning duplicates in the machine learning and database communities. Most research efforts have focused on automatic approaches rather than interactive support. Traditional approaches make use of only attribute information, where entities are matched based on the values of their attributes. Most work primarily focused on defining approximate string matching algorithms [31][1][10] and fuzzy matching [22]. Other attribute-based approaches have been adaptive [16][30][9][24]. Recently a number of approaches have been developed which make use of relational information to help in the resolution process [15][14][21]. In these approaches, the relational graph is taken into account for finding potential duplicates. Within the database community, there has been some research on interactive data cleaning [26][27] which typically focuses on a single table, but little work has been done to combine visual and analytic information effectively with interactive tool for entity resolution tasks.

There also have been many efforts on resolving the geospatial entities in the GIS community especially for the data integration. Commercial geographic-information systems [20] use spatial coordinates to join location references. One-sided nearest join is the most common approach, where location's references $\boldsymbol{l_i} \in \boldsymbol{A}$ and $\boldsymbol{l_j} \in \boldsymbol{B}$ are labeled as duplicates, if $\boldsymbol{l_j}$ is closest to $\boldsymbol{l_i}$ given all locations in $\boldsymbol{B}$. The asymmetry in the above approach is addressed by Beeri et al. [6]. They modify the definition by adding an extra condition that $\boldsymbol{l_i}$ is also closest to $\boldsymbol{l_j}$ given all locations in $\boldsymbol{A}$. They show that the addition of this condition may help improve precision of results, though the strong conditions may compromise recall. Another recent approach [19] integrates heterogeneous geo-referenced data available from the web. Here, they match geospatial objects by comparing their attributes using string matching algorithms. Other methods for geospatial data integration such as [7][33][12][3][4] integrate digital maps. Image processing techniques [12] have also been applied in merging digital satellite image maps. Though these methods help in a global alignment of geospatial database, the efficacy of these methods strongly depends upon the quality and richness of the satellite image.

3. GeoDDupe Interface

In this section, we illustrate the interface using a sample geospatial dataset and describe the overall entity resolution process. The dataset consists of two data source files taken from the Geographic Names Database of the United States Board on Geographic Names maintained by the National Geospatial-Intelligence Agency. The source files contain 1,116 locations and 28,853 locations in Iraq respectively. For each location, the neighbor locations are chosen such that the distance between them is less than 50km and linked with undirected edges. 2,766 edges and 268,759 edges are identified from each file respectively.

Figure 1 shows the overall *GeoDDupe* interface, which is composed of three coordinated windows [8]: the potential duplicate viewer, the relational context viewer, and the data detail viewer. The potential duplicate viewer (on the left) shows a list of potential duplicate location pairs that are identified and ranked based on the user-defined similarity metrics. Users can select a potential duplicate pair, and then the relational context viewer (on the upper right corner) visualizes the neighborhood relationship between the potential duplicate location pair in various ways. Figure 1 shows a simple semantic graph layout which represents the neighborhood relationship between the two potential duplicate locations, "Sian" and "Sian" by placing the shared neighboring locations in-between them and non-shared neighboring locations on the sides.. Finally the data detail viewer (in the lower right corner) shows all the attribute values of locations displayed in the relational context viewer.

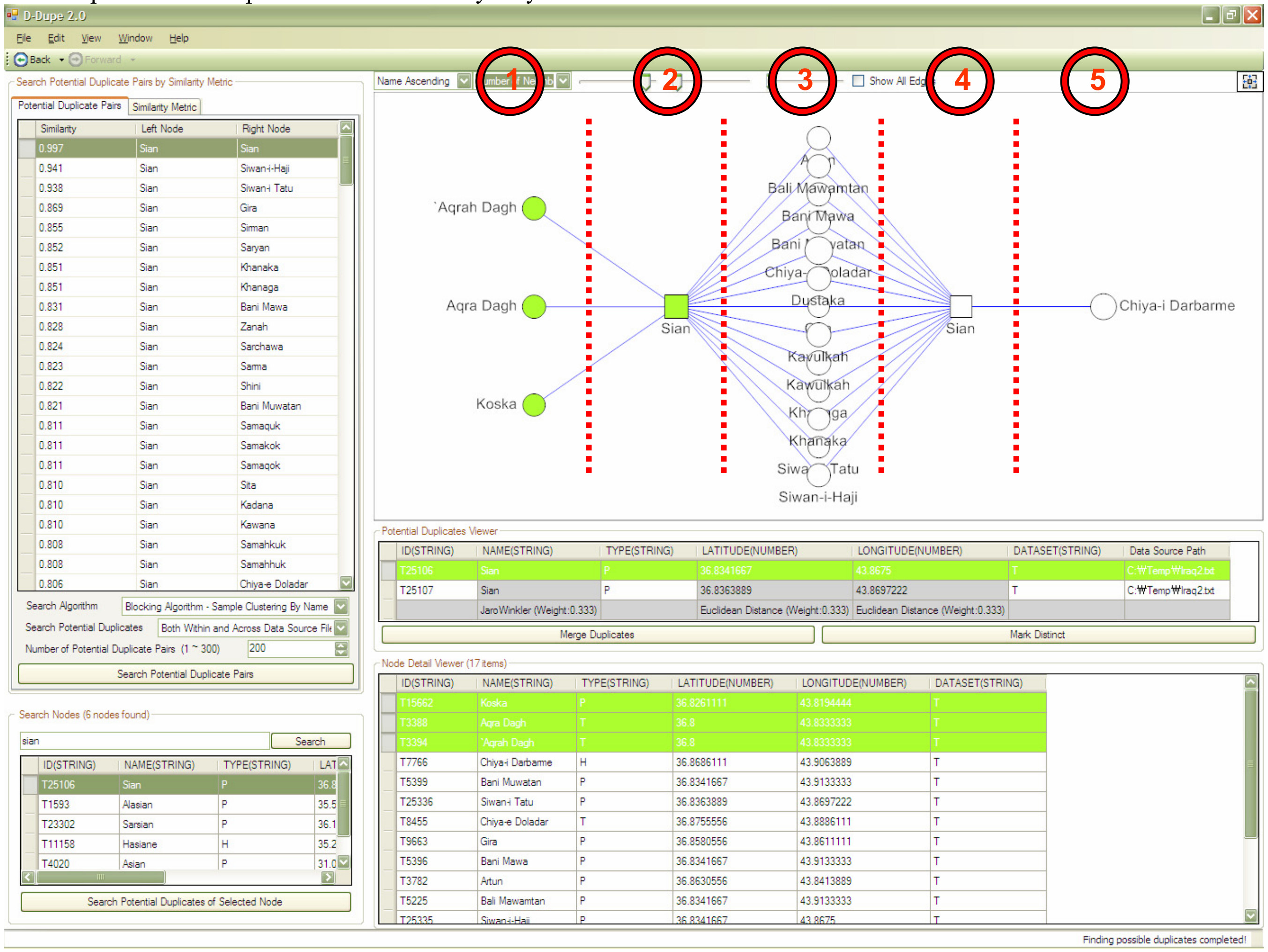
Users begin the entity resolution process by loading one or more geospatial data files depending on their tasks (deduplication or integration). Before starting potential duplicate search, users need to define similarity metrics which describe how similarity between two locations should be computed. For example, if users consider locations to be similar when they have similar names and similar geospatial positions, users can select three location attributes: location name, latitude, and longitude, for computing the location similarity. Once the similarity metric is defined, users can have *GeoDDupe* search for the potential duplicate locations by pressing the search button in the potential duplicate viewer. *GeoDDupe* provides a few options which enable users to control the performance and completeness of search depending upon their tasks.

When the search has been completed, users can see the top *N* (by default 200) potential duplicate location pairs in the potential duplicate viewer. Users can scroll through the list of potential duplicate pairs and select a potential duplicate pair for analysis. Users can then see the neighborhood relationship between the potential duplicate locations in the relational context viewer. The viewer is designed to show a sub-network which only contains the potential duplicate locations and their neighboring locations. In the semantic layout mode, the viewer is divided into five regions (red dotted lines in Figure 1). The potential duplicate locations are located in the second and fourth regions, the shared neighboring locations in the third region, and the non-shared neighboring locations in the first and fifth regions. In Figure 1, the potential duplicate locations have 13 shared neighboring locations, and three and one non-shared neighboring locations respectively, which means it is highly likely that they are duplicates in the dataset. Users can resize, sort, filter, and brush the nodes and edges in the viewer for better understanding of the relationship between the potential duplicate locations.

Users can resolve the potential duplicate locations in three ways: 1) merge, 2) mark as distinct, 3) ignore for later or other users' decision. If duplicate locations are merged, *GeoDDupe* automatically searches for the next possible duplicate location for the merged location and shows it in the relational context viewer. With this feature, users can make a series of resolution decisions for the multiple references to the same location and resolve them incrementally. In the following subsections, the capabilities supported in each viewer and underlying design concepts are explored in more detail.

3.1. Potential Duplicate Viewer

The potential duplicate viewer window consists of two sub-panels; "Search Potential Duplicate Pairs by Similarity Metric" panel on the top and "Search Node by Keyword" panel on the bottom (left in Figure 1). The "Search Potential Duplicate Pairs by Similarity Metric" panel allows users to define a similarity metric appropriate for their specific tasks and search for the potential duplicate locations from the imported dataset. The "Similarity Metric" tab supports two types of similarity: attribute similarity and relation similarity. In the attribute similarity group box, users can see all the location attributes available in the dataset. Figure 2(a) shows that the dataset contains 6 location attributes: location id, location name, location type, longitude, latitude, and dataset (first column). The "Similarity Measure" column (second column) allows users to choose a similarity measure function for each attribute. Each cell in this column displays different similarity function options depending on the attribute data type (e.g. string or number).



	Similarity	Left Node	Right Node
	0.997	Sian	Sian
	0.941	Sian	Siwan-i-Haji
	0.938	Sian	Siwan-i Tatu
	0.869	Sian	Gira
	0.855	Sian	Siman
	0.852	Sian	Saryan
	0.851	Sian	Khanaka
	0.851	Sian	Khanaga
	0.831	Sian	Bani Mawa
	0.828	Sian	Zanah
	0.824	Sian	Sarchawa
	0.823	Sian	Sama
	0.822	Sian	Shini
	0.821	Sian	Bani Muwatan
	0.811	Sian	Samaquk
	0.811	Sian	Samakok
	0.811	Sian	Samaqok
	0.810	Sian	Sita
	0.810	Sian	Kadana
	0.810	Sian	Kawana
	0.808	Sian	Samahkuk
	0.808	Sian	Samahhuk
	0.806	Sian	Chiya-e Doladar

Search Algorithm: Blocking Algorithm - Sample Clustering By Name
Search Potential Duplicates: Both Within and Across Data Source Fil
Number of Potential Duplicate Pairs (1 ~ 300): 200

sian

	ID(STRING)	NAME(STRING)	TYPE(STRING)	LAT
	T25106	Sian	P	36.8
	T1593	Alasian	P	35.5
	T23302	Sarsian	P	36.1
	T11158	Hasiane	H	35.2
	T4020	Asian	P	31.0

	ID(STRING)	NAME(STRING)	TYPE(STRING)	LATITUDE(NUMBER)	LONGITUDE(NUMBER)	DATASET(STRING)	Data Source Path
	T25106	Sian	P	36.8341667	43.8675	T	C:₩Temp₩Iraq2.txt
	T25107	Sian	P	36.8363889	43.8697222	T	C:₩Temp₩Iraq2.txt
		JaroWinkler (Weight:0.333)		Euclidean Distance (Weight:0.333)	Euclidean Distance (Weight:0.333)		

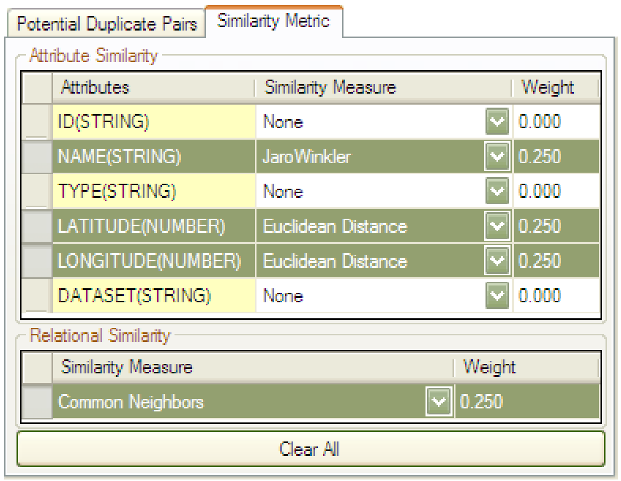
	ID(STRING)	NAME(STRING)	TYPE(STRING)	LATITUDE(NUMBER)	LONGITUDE(NUMBER)	DATASET(STRING)
	T15662	Koska	P	36.8261111	43.8194444	T
	T3388	Aqra Dagh	T	36.8	43.8333333	T
	T3394	`Aqrah Dagh	T	36.8	43.8333333	T
	T7766	Chiya-i Darbarme	H	36.8686111	43.9063889	T
	T5399	Bani Muwatan	P	36.8341667	43.9133333	T
	T25336	Siwan-i Tatu	P	36.8363889	43.8697222	T
	T8455	Chiya-e Doladar	T	36.8755556	43.8886111	T
	T9663	Gira	P	36.8580556	43.8611111	T
	T5396	Bani Mawa	P	36.8341667	43.9133333	T
	T3782	Artun	P	36.8630556	43.8413889	T
	T5225	Bali Mawamtan	P	36.8341667	43.9133333	T
	T25335	Siwan-i-Haji	P	36.8341667	43.8675	T

Figure 1 *GeoDDupe* consists of three coordinated windows; potential duplicate viewer on the left, relational context viewer on the upper right corner, and data detail viewer on the lower right corner. The potential duplicate viewer shows a list of potential duplicate location pairs that were identified based on the user-defined similarity metrics. The relational context viewer visualizes the neighborhood relationship of the potential duplicate location pair selected in the potential duplicate viewer. The data detail viewer shows all the attribute values of the locations displayed in the relational context viewer. Both viewers are tightly coupled.

For example, users can use string match functions such as Levenstein, Jaccard, Jaro, JaroWinkler, MongeElkan, etc. for computing the name similarity. Each similarity measure function has its advantages; for example, Levenstein is able to capture misspellings, JaroWinkler performs well for matching person names and abbreviations. The choice of similarity measure function depends upon the application. For example in [28], we found that Levenstein works best for matching geographical names. On the other hand, users can apply numerical distance function such as Euclidean Distance to the longitude and the latitude attributes for computing the position similarity. In the weight column (third column), users can assign either positive or negative weights to the corresponding similarity functions. In addition to the attribute similarity, users can also make use of relational information in searching for the potential duplicate locations by analyzing the commonness of neighborhoods with relational similarity measure functions such as *Common Neighbors* and *Higher-Order Neighborhoods* [14] in the relational similarity group box.

The "Potential Duplicate Pairs" tab shows the list of potential duplicate location pairs identified based on the similarity metric defined as above (Figure 2(b)). In this tab, users can set three search options to control the search process. First, users can choose different search strategies. By default, *GeoDDupe* compares all the possible pairs of locations to find the potential duplicate locations. However, the exhaustive search in this example (i.e. comparing all pairs of 30,000 locations) needs about half billion comparisons, which is clearly impractical. In order to deal with this performance issue, *GeoDDupe* is designed to support *blocking* algorithms. *Blocking* algorithms [17] generate candidate matches and partition the full cross product of location comparisons into mutually exclusive blocks as a preprocessing step. This can significantly improve the search performance. Second, users can choose a way of picking comparing locations in the multiple data files. For example, if users want to resolve duplicates within the same data file, they can make *GeoDDupe* search the candidate locations only within the same file by choosing the "Within Data Source File (Deduplication)" option. On the other hand, if users want to integrate geospatial datasets by matching the locations across multiple data files, they can choose the "Across Data Source Files (Integration)" option. By default, the "Both Within and Across Data Source Files" option is selected to support both tasks at once. Finally, users can specify the number of potential duplicate pairs to be displayed in the list so that they can only focus on top *N* potential duplicate pairs from the search results. The search process can be stopped at any time by pressing the stop button (that was the search button when starting search). *GeoDDupe* then shows only the partial search results identified until the process is stopped.

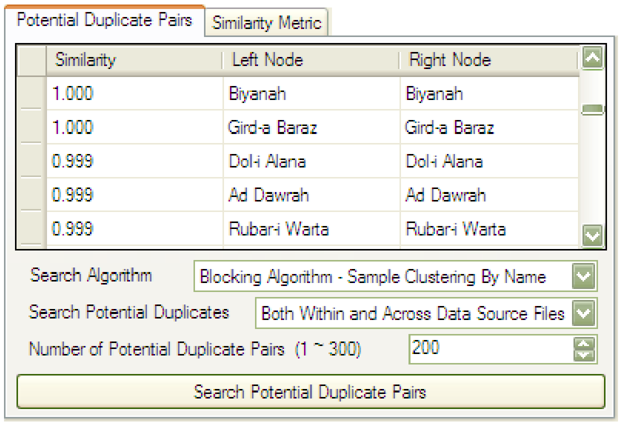
Sometimes users may need to search for a specific location containing certain keywords in their attribute values rather than search all potential duplicate pairs with the similarity metric. The "Search Location by Keyword" panel located on the bottom of the potential duplicate viewer supports the keyword search for single locations. If users want to check if there exist any locations similar to a single location in the list, they can easily find its potential duplicate locations just by double-clicking on it.



Attributes	Similarity Measure	Weight
ID(STRING)	None	0.000
NAME(STRING)	JaroWinkler	0.250
TYPE(STRING)	None	0.000
LATITUDE(NUMBER)	Euclidean Distance	0.250
LONGITUDE(NUMBER)	Euclidean Distance	0.250
DATASET(STRING)	None	0.000

Similarity Measure	Weight
Common Neighbors	0.250

(a) The "Similarity Metric" tab allows users to use attribute or relational similarity measures for computing the location similarity



Similarity	Left Node	Right Node
1.000	Biyanah	Biyanah
1.000	Gird-a Baraz	Gird-a Baraz
0.999	Dol-i Alana	Dol-i Alana
0.999	Ad Dawrah	Ad Dawrah
0.999	Rubar-i Warta	Rubar-i Warta

Search Algorithm: Blocking Algorithm - Sample Clustering By Name
Search Potential Duplicates: Both Within and Across Data Source Files
Number of Potential Duplicate Pairs (1 ~ 300): 200

(b) The "Potential Duplicate Pairs" tab shows the identified potential duplicate pairs ranked by similarity

Figure 2 Potential Duplicate Viewer

3.2. Relational Context Viewer

Instead of visualizing the whole geospatial network, *GeoDDupe* shows only the sub-network relevant for the entity resolution task with a novel semantic layout. This simplification not only reduces the users' cognitive workload in scanning the network structure but also allows *GeoDDupe* to scale to large networks. In Figure 3(a), the potential duplicate locations are represented as squares and placed in the middle of screen and the shared neighbor locations of those duplicates are placed in-between them, and non-shared neighbor locations are placed on the sides. By default only the links between the potential duplicate locations and their neighboring locations are shown. However, the links among the neighboring locations can also be displayed by checking the “Show All Edges” checkbox in the toolbar. With this simplified layout, users can easily recognize the neighborhood relation between the potential duplicate locations. The more neighboring locations the potential duplicates share, the more likely the duplicates refer to the same location in real world.

While the semantic layout facilitates quick understanding of the neighborhood structure between the potential duplicates, it is not easy for users to understand the relative positions of the neighboring locations in real world, which may be useful for the resolution decisions. Therefore, the relational context viewer is designed to support an alternative layout to represent locations’ spatial information as well (Figure 3(b)). In this layout, users can see the relative spatial positions of the neighboring locations in 2D space as well as the neighborhood structure with the edges connecting neighboring locations. Either of two layouts can be transformed to the other with animation, which allows users to easily recognize the relationship and differences between two layouts.

If the potential duplicate locations are found across different datasets (for data integration), it is not unusual that they don’t have any shared neighboring locations since their neighboring locations are also from the different datasets. Figure 4(a) shows a semantic graph layout which doesn’t have any shared neighboring locations between the potential duplicate locations. This layout may misrepresent the actual neighborhood relationship of the potential duplicate locations because there may exist unidentified duplicate locations across the non-shared neighboring locations of each potential duplicate location. To protect users from this kind of misinterpretation, the relational context viewer provides a similarity threshold slider in the toolbar so that users can check if there are any potential duplicate locations across the non-shared neighbors. If any location pairs across the non-shared neighboring locations have higher similarity than the threshold specified by the slider, they are automatically moved in between the potential duplicate locations and placed next to each other as shown in Figure 4(b). With this transformed layout, users can not only anticipate the possible matches across the non-shared neighboring locations, but also understand the actual neighborhood relationship more clearly.

Even though the relational context viewer provides users useful visual information about the neighboring locations either by semantic layout or by geo-spatial layout, users may still need to examine them on a real world map for better resolution decisions. *GeoDDupe* enables users to export the locations in the relational context viewer to the external Geographic Information System such as Google Earth (earth.google.com) as shown in Figure 5. By comparing locations both in the semantic layout and on the actual map at once, users can be more confident about the resolution decisions they make.

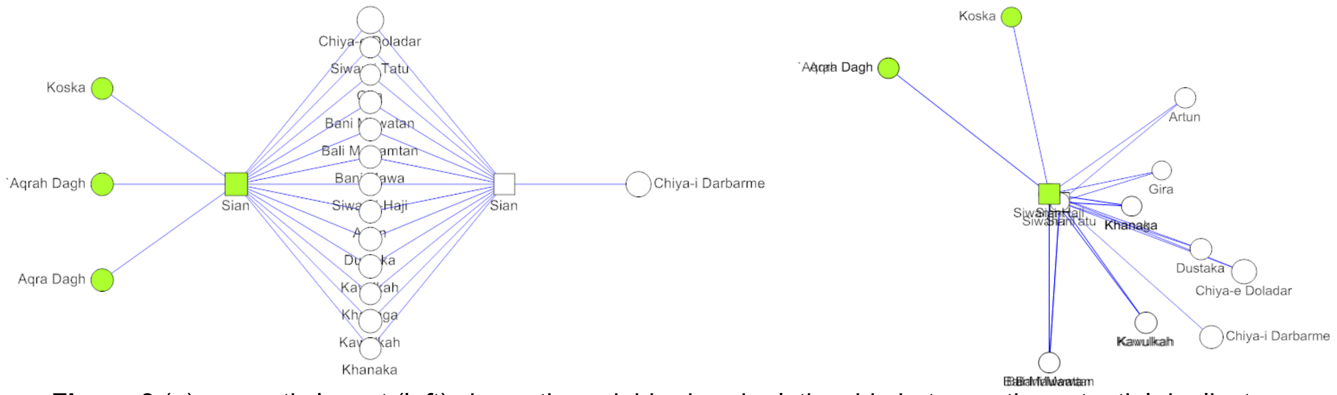


Figure 3 (a) semantic layout (left) shows the neighborhood relationship between the potential duplicates (b) spatial layout (right) represents the relative spatial positions of neighboring locations on 2D space

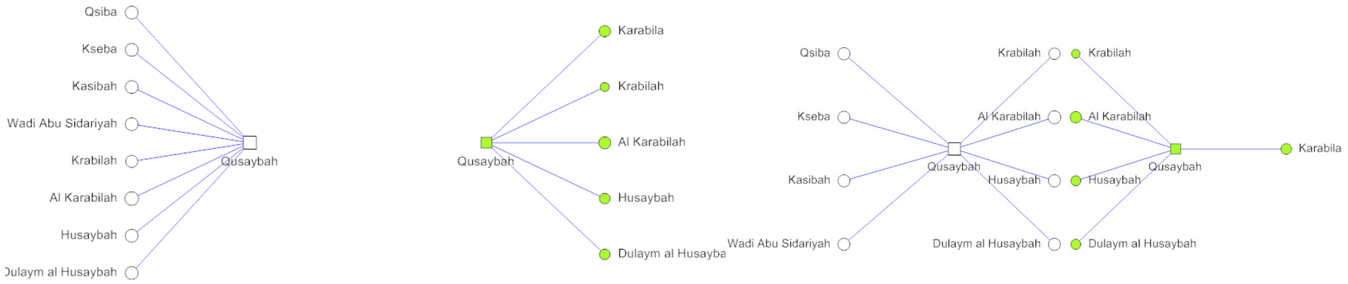
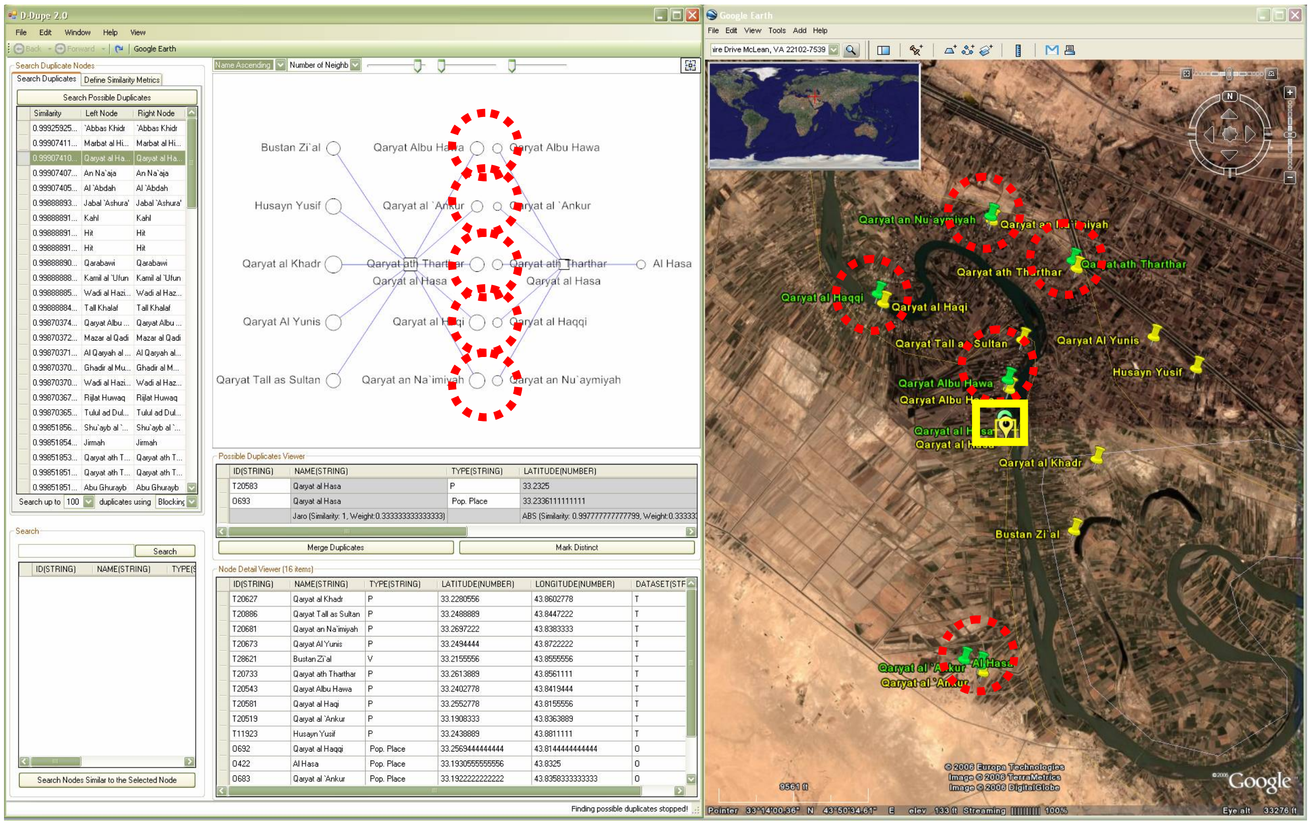


Figure 4 (a) original semantic layout (left) for the potential duplicates identified across datasets (b) transformed layout (right) where the matching pairs identified across non-shared neighbors are placed in the middle



Similarity	Left Node	Right Node
0.99925925...	`Abbas Khidr	`Abbas Khidr
0.99907411...	Marbat al Hi...	Marbat al Hi...
0.99907410...	Qaryat al Ha...	Qaryat al Ha...
0.99907407...	An Na`aja	An Na`aja
0.99907405...	Al `Abdah	Al `Abdah
0.99888893...	Jabal `Ashura'	Jabal `Ashura'
0.99888891...	Kahl	Kahl
0.99888891...	Hit	Hit
0.99888891...	Hit	Hit
0.99888890...	Qarabawi	Qarabawi
0.99888888...	Kamil al `Ufun	Kamil al `Ufun
0.99888885...	Wadi al Hazi...	Wadi al Haz...
0.99888884...	Tall Khalaf	Tall Khalaf
0.99870374...	Qaryat Albu ...	Qaryat Albu ...
0.99870372...	Mazar al Qadi	Mazar al Qadi
0.99870371...	Al Qaryah al ...	Al Qaryah al...
0.99870370...	Ghadir al Mu...	Ghadir al M...
0.99870370...	Wadi al Hazi...	Wadi al Haz...
0.99870367...	Rijlat Huwaq	Rijlat Huwaq
0.99870365...	Tulul ad Dul...	Tulul ad Dul...
0.99851856...	Shu`ayb al `...	Shu`ayb al `...
0.99851854...	Jirmah	Jirmah
0.99851853...	Qaryat ath T...	Qaryat ath T...
0.99851851...	Qaryat ath T...	Qaryat ath T...
0.99851851...	Abu Ghurayb	Abu Ghurayb

Search up to 100 duplicates using Blocking

ID(STRING)	NAME(STRING)	TYPE(

Possible Duplicates Viewer

ID(STRING)	NAME(STRING)	TYPE(STRING)	LATITUDE(NUMBER)
T20583	Qaryat al Hasa	P	33.2325
0693	Qaryat al Hasa	Pop. Place	33.2336111111111
	Jaro (Similarity: 1, Weight:0.333333333333333)		ABS (Similarity: 0.997777777777799, Weight:0.33333

Merge Duplicates | Mark Distinct

Node Detail Viewer (16 items)

ID(STRING)	NAME(STRING)	TYPE(STRING)	LATITUDE(NUMBER)	LONGITUDE(NUMBER)	DATASET(STF
T20627	Qaryat al Khadr	P	33.2280556	43.8602778	T
T20886	Qaryat Tall as Sultan	P	33.2488889	43.8447222	T
T20681	Qaryat an Na`imiyah	P	33.2697222	43.8383333	T
T20673	Qaryat Al Yunis	P	33.2494444	43.8722222	T
T28621	Bustan Zi`al	V	33.2155556	43.8555556	T
T20733	Qaryat ath Tharthar	P	33.2613889	43.8561111	T
T20543	Qaryat Albu Hawa	P	33.2402778	43.8419444	T
T20581	Qaryat al Haqi	P	33.2552778	43.8155556	T
T20519	Qaryat al `Ankur	P	33.1908333	43.8363889	T
T11923	Husayn Yusif	P	33.2438889	43.8811111	T
0692	Qaryat al Haqqi	Pop. Place	33.2569444444444	43.8144444444444	0
0422	Al Hasa	Pop. Place	33.1930555555556	43.8325	0
0683	Qaryat al `Ankur	Pop. Place	33.1922222222222	43.8358333333333	0

Finding possible duplicates stopped!

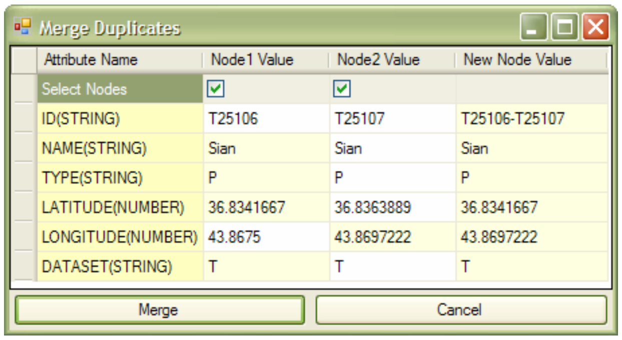
Figure 5 The locations displayed in the relational context viewer can be exported and projected onto Google Earth. In Google Earth, the potential duplicate locations are placed in the center (yellow rectangle), and 5 other potential duplicate location pairs identified in the relational context viewer (red dotted circles on the relational context viewer) are scattered over the map (red dotted circles on the map)

3.3. Data Detail Viewer

The data detail viewer shows all the attribute values of the locations visualized in the relational context viewer. The data detail viewer consists of three sub-panels: the potential duplicate viewer, the node viewer, and the edge viewer (Figure 1). The data detail viewer is tightly-coupled with the relational context viewer so that users can choose locations in the layout and observe the corresponding attribute values in the data detail viewer, and vice versa.

In the potential duplicate viewer, users can either merge the currently listed potential duplicate locations or mark them as distinct locations. When the "Merge Duplicates" button is pressed, the "Merge Duplicates" dialog box pops up. The dialog box shows all the location attribute names in the first column and corresponding attribute values of the potential duplicate locations in the next two columns (Figure 6). The last column shows the attribute values of the location to be created by merging the potential duplicate locations. Users can either pick values from the duplicate locations' cells or type in new values to the cells in the last column. When the "Merge" button is pressed, *GeoDDupe* creates a new location with the assigned attribute values and replaces all the potential duplicate locations in the dataset with it.

If the potential duplicate locations are not actual duplicates, users can also mark them as distinct locations using the "Mark Distinct" button in the potential duplicate viewer. Then, the marked pairs are automatically excluded from the search results, but they are maintained internally for later revision. On the other hand, if users have made an incorrect merging decision, they can restore the prior locations as well by splitting the merged location.



Attribute Name	Node1 Value	Node2 Value	New Node Value
Select Nodes	✓	✓	
ID(STRING)	T25106	T25107	T25106-T25107
NAME(STRING)	Sian	Sian	Sian
TYPE(STRING)	P	P	P
LATITUDE(NUMBER)	36.8341667	36.8363889	36.8341667
LONGITUDE(NUMBER)	43.8675	43.8697222	43.8697222
DATASET(STRING)	T	T	T

Figure 6 "Merge Duplicates" dialog window

4. System Architecture

GeoDDupe was written in C# using the University of Maryland's open source toolkit, Piccolo[5], for network visualization. *GeoDDupe* supports both text file and MS Access database for its input. It can be connected to many relational databases through ODBC connection as long as they fit the basic data schema that is represented as Boyce-Codd normal form (that consists of two tables representing two individual entities and one table for their inter-relationship) or a standard node-link graph model.

The *GeoDDupe* architecture is basically composed of three parts: Model, View and Controller [11]. The *GeoDDupe* Model manages all the data related processes as well as data mining algorithms so that *GeoDDupe* View can just request the necessary data from the Model and visualize them without caring about the internal data structures or algorithms that the Model uses. The *GeoDDupe* Model consists of three modules: the data mining algorithm module, the graph data structure module, and the data input/output module. They are designed to be modular so that other developers can easily add their own domain-specific algorithms and similarity measures to improve accuracy and performance for entity resolution, or add other IO modules to support various types of databases such as XML DB and OODB. The current data mining algorithm module uses an open source library SimMetrics [25] for string distance measure functions. The *GeoDDupe* Controller manages all the events raised by the viewers, coordinates the viewers, and controls all the dataflow among the viewers. In addition, it manages the history mechanism so that users can switch back and forth to the previously searched sets of potential duplicates without re-searching the same set repeatedly, which often slows down the entity resolution process.

5. Case Study

Two real-world datasets were used for the case study. The first dataset represents all names of locations in Afghanistan taken from the Geographic Names Database (http://earth-info.nga.mil/gns/html/cntry_files.html) of the US Board on Geographic Names maintained by the National Geospatial-Intelligence Agency. The Geographic Names Database is the official standard for spelling of foreign place names for use throughout the US Government. The first dataset contains 202,210 records of locations (Jan 2007). We preprocessed the dataset and connected the neighbor locations with undirected edges to represent the neighborhood relationship. A total of 1,262,512 edges were generated by connecting the locations if the distance between them is less than 25km. The second database was prepared by the UK Permanent Committee on Geographic Names. It contains 2,096 records of locations in Helmand Province, Afghanistan. 2,037 edges were generated using the same method. Both datasets were cleaned before published on the web. Each location in the datasets has a name (in the BGN-style Romanization format), spatial coordinates (latitude and longitude), and location type. The ground truth data covering the actual mappings between two datasets was prepared by the US BGN and UK Permanent Committee on Geographic Names, and it covers most of the locations in the second dataset. The ground-truth data contains a total of 2006 pairs of matching locations.

We imported both datasets to *GeoDDupe* and defined the similarity metric using three location attributes: location name, longitude, and latitude. We applied the JaroWinkler similarity measure function to the location name and the Euclidean Distance similarity measure function to the pair of longitude and the latitude respectively. Equivalent weights were assigned to each similarity measure function. For search options, we used a very simple *blocking* algorithm for fast search, which clusters the locations based on the first four digits of latitude. Since we focused more on data integration task, we selected the "Across Data Source Files (Integration)" option so that *GeoDDupe* picks comparing locations only across the data files. For the experiment, we used a Windows PC (Windows XP pro SP2) with 3.6GHz Pentium 4 CPU, 2.5GB main memory, and 128MB graphic memory with 1400 by 1050 screen resolution.

In less than a minute, *GeoDDupe* identified 300 potential duplicate locations with the lowest similarity value 0.978 from the datasets. We used the similarity slider and set the similarity threshold to 0.95 to detect the potential duplicates across the non-shared neighboring locations. In this case study, *w*e were able to identify more than 200 matching locations across datasets and integrate 135 locations within a half hour of use. Among 135 integrated locations, 133 locations were confirmed as matching locations from the ground truth data and two locations seemed to be the matching locations which were not covered by the ground truth data. In addition, although both datasets were carefully prepared for public release, we were able to detect 190 and 18 duplicate locations from each dataset respectively, and deduplicate them during data integration.

Conclusions

In this paper, we introduced *GeoDDupe,* an interactive tool for entity resolution in geospatial data. *GeoDDupe* integrates data mining algorithms with an interactive information visualization interface to support geospatial entity resolution tasks. *GeoDDupe* is designed to deal with both spatial and non-spatial data while making use of the relational information of locations for geospatial entity resolution. *GeoDDupe* provides sophisticated and automated entity resolution algorithms which enables users to easily discover the potential duplicate locations from the geospatial datasets. In addition, *GeoDDupe* provides a simple, stable, and meaningful network visualization which guides users to make accurate exploration and resolution decisions. More formative user studies are required to verify the efficacy of *GeoDDupe* approach for geospatial entity resolution, but the utility of *GeoDDupe* seems promising. Finally, *GeoDDupe* demonstrates the impact of combining data mining techniques with novel information visualization for data exploration and analysis tasks.

References

[1] A. E. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 267-270, 1996.

[2] A. McCallum and B. Wellner. Conditional models of identity uncertainity with application to noun conference. In *Proceedings of Neural Information Processing Systems Conference*, 2004.

[3] A. Saafeld. Conflation-automated map compilation. *International Journal of Geographical Information Systems*, 2(3):217-228, 1988.

[4] A. Samal, S. Seth, and K. Cueto. A feature based approach to conflation of geospatial sources. *International Journal of Geographical Information Systems*, 18(00):1-31, 2004.

[5] B. Bederson, J. Grosjea, and J. Meyer. Toolkit Design for Interactive Structured Graphics. *IEEE Transactions on Software Engineering*. 30(8): 535-546, 2004.

[6] C. Beeri, Y. Kanza, E. Safra, and Y. Sagiv. Object fusion in geographic information systems. In *Proceedings of International Conference on Very Large Data Bases*, 2004.

[7] C. Chen, C. A. Knoblock, C. Shahabi, Y. Chiang, and S. Thakkar. Automatically and accurately conflating orthoimagery and street maps. In *Proceedings of the 12th ACM International Symposium on Advances in Geographic Information Systems* (ACM-GIS'04), 2004.

[8] C. North and B. Shneiderman. A taxonomy of multiple window coordinations. Technical Report CS-TR-3854, University of Maryland, 1997.

[9] E. S. Ristad and P. N. Yianilos. Learning string-edit distance. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 20(5): 522-532, 1998.

[10] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31.88, 2001.

[11] G.E. Krasner, S.T. Pope, A Cookbook for Using The Model-View-Controller User Interface Paradigm in Smalltalk-80, *Journal of Object-Oriented Programming*, 1(3), 26 - 49, 1988.

[12] H. Mayer. Automatic object extraction from aerial imagery - a survey focusing on buildings. *Computer Vision and Image Understanding*, 74(2), 1999.

[13] H. Newcombe, J. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. *Science*, 130:954-959.

[14] I. Bhattacharya and L. Getoor. Entity Resolution in Graphs, *Mining Graph Data* (Lawrence B. Holder and Diane J. Cook, eds.), Wiley, 2006.

[15] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *Proceedings of SIGMOD Workshop on Data Mining and Knowledge Discovery*.

[16] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of ACM SIGKDD International. Conference on Knowledge Discovery and Data Mining*, 39-48, 2003.

[17] M. Bilenko, B. Kamath, R. J. Mooney. Adaptive Blocking: Learning to Scale Up Record Linkage, In *Proceedings of International Conference on Data Mining*, 87-96, 2006.

[18] M. Bilgic, L. Licamele, L. Getoor, and B. Shneiderman. D-Dupe: An Interactive Tool for Entity Resolution in Social Networks, *IEEE Symposium on Visual Analytics Science and Technology*, 2006.

[19] M. Michalowski, J. L. Ambite, S. Thakkar, R. Tuchinda, C. A. Knoblock, and S. Minton. Retrieving and semantically integrating heterogeneous data from the web. *IEEE Intelligent Systems*, 19(3), 2004.

[20] M. Minami. Using Arcmap, 2000. Enviornmental Systems Research Institute, Inc.

[21] P. Singla and P. Domingos. Multi-relational record linkage. In *Proceedings of ACM SIGKDD Workshop on Multi-Relational Data Mining*, 2004.

[22] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 313-324, 2003.

[23] S. Lawrence, K. Bollacker, and C. L. Giles. Autonomous citation matching. In *Proceedings of the Third International Conference on Autonomous Agents*, 1999.

[24] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 269-278, 2002.

[25] SimMetrics: Open Source Similarity Measure Library (http://www.dcs.shef.ac.uk/~sam/simmetrics.html)

[26] T. Dasu and T. Johnson. Exploratory Data Mining and Data Cleaning. John Wiley & Sons, Inc., 2003.

[27] V. Raman and J. Hellerstein. Potter's wheel: An interactive data cleaning system. In *Proceedings of International Conference on Very Large Data Bases*, 381-390, 2001.

[28] V. Sehgal, L. Getoor, and P. Viechnicki. Entity Resolution in Geospatial Data Integration, In *Proceedings of 14th International Symposium on Advances in Geographic Information Systems*, November 2006

[29] W. E. Winkler. Methods for record linkage and Bayesian networks. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 2002.

[30] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 475-480, 2002.

[31] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of IJCAI Workshop on Information Integration on the Web*, 73-78, 2003.

[32] X. Li, P. Morie, and D. Roth. Semantic integration in text: From ambiguous names to identifiable entities. *AI Magazine Special Issue on Semantic Integration*, 2005.

[33] Y. Doytsher and S. Filin. The detection of corresponding objects in a linear-based map confliation. Surveying and Land Information Systems, 60(2):117-128, 2000.