2010

Information integration for graph databases

Ee Peng LIM
Singapore Management University, eplim@smu.edu.sg

Aixin SUN

Anwitaman DATTA

CHANG KUIYU

DOI: https://doi.org/10.1007/978-1-4419-6515-8_10

Chapter 10
Information Integration for Graph Databases

Ee-Peng Lim, Aixin Sun, Anwitaman Datta, and Kuiyu Chang

Abstract With increasing interest in querying and analyzing graph data from multiple sources, algorithms and tools to integrate different graphs become very important. Integration of graphs can take place at the schema and instance levels. While links among graph nodes pose additional challenges to graph information integration, they can also serve as useful features for matching nodes representing real-world entities. This chapter introduces a general framework to perform graph information integration. It then gives an overview of the state-of-the-art research and tools in graph information integration.

10.1 Introduction

Graph is fast becoming an important data genre in today's database and data analysis systems and applications. Web itself is a very large graph with Web pages as nodes and links among them as edges. The Internet that makes Web possible is also a large graph with computers as nodes and network links as edges. The emergence of Web 2.0 applications further creates many other graphs that associate Web users with one another, and graphs that associate Web users with Web objects including photos (e.g., Flickr[1]), videos (e.g., Youtube[2]), and questions/answers (e.g., Yahoo! Answers[3]).

Graph is often used for data modeling or knowledge representation. Entity relationship model [6] is essentially a graph consisting of entity types and relationship types as nodes and connections among them as edges. It is widely used to design databases conceptually before the relational schemas are created. Ontology is a another kind of graph used for sharing knowledge between applications from the same domain [11]. Instead of keeping schema and data separate as in

E.-P. Lim (✉)
School of Information Systems, Singapore Management University, Singapore
e-mail: eplim@smu.edu.sg

[1] http://www.flickr.com.
[2] http://www.youtube.com
[3] http://answers.yahoo.com.

database design, ontology uses nodes to represent both schema and data objects and edges to represent schema–level relationships (e.g., PhDStudent is a subclass of Graduate student), schema–data relationships (e.g., John and Mary are instances of PhDStudent), and data–level relationships (e.g., John is a friend of Mary). With ontology graphs defined for different knowledge domains, knowledge-based systems and applications are expected to interoperate using the same set of concepts to describe data and to perform reasoning on them. Ontology is later adopted by Semantic Web as an extension to the existing Web enabling machines to standardize the description of Web objects and services so as to understand them and deploy Web services.

Graphs also exist in relational databases although the graph structures in relational database are very much *normalized*. Records in a relational table can be associated with records from another table using foreign key references or via a relation containing many-to-many associations between record keys. For example, Internet Movie Database (IMDb) is an online relational database of movie, actor, and director-related data. These movie-related entities essentially form a graph if we view them as nodes and associations among them as edges. For example, every movie record is linked to its actors and director, and each actor is linked to all of his or her movies. Compared to ontology, graphs that are embedded in relational data are more structured and their structures are governed by the database schema.

Other than the above two schools of thought, schemaless graphs with nodes and edges assigned labels have also been widely used to represent complex structures such as protein and chemical compounds. Examples of such graph models are OEM [5, 10, 20]. With the popularity of e-commerce, such graph models have also been applied to modeling XML data so as to formulate and evaluate queries on XML databases.

Information integration [13], sometimes also known as *data integration* and *semantic integration*, refers to merging information from different data sources in order to gain a more complete set of data for developing new applications, and for conducting data analysis/data mining. The new applications to be developed can be due to the demand for new functionalities or due to application, database, or even enterprise level merger activities. Since the original databases residing at different data sources are likely administered by different parties, information integration has to address three major technical challenges arising from data heterogeneity [18, 21, 22]:

– *Schema-Level Heterogeneity*: This refers to schema differences between two or more databases to be integrated. Depending on the physical data models of the original data sources, the scheme level heterogeneity may involve matching schema elements of original databases and mapping them to the schema elements of an integrated schema [22]. In the case of schemaless graph data, schema-level heterogeneity does not exist as there is only a single node type and a single edge type.

– *Instance-Level Heterogeneity*: Data heterogeneity occurs at the instance level when data instances from different data sources but describing the same real-world entity do not look alike. The conflicts incurred can be classified into *entity identity conflicts* [16] and *attribute value conflicts* [17]. Entity identity conflicts refer to the difficulties in resolving records that model the same real-world entities due to synonym and homonym problems [16]. Attribute value conflicts refer to attribute value differences in records to be integrated together. The tasks of resolving entity identity and attribute value conflicts are known as *entity identification* (or *entity resolution*) and *attribute value conflict resolution*, respectively.
– *Federated Query Processing*: Heterogeneous databases can be integrated either physically or virtually. The former refers to migrating the original data to a central database to be physically merged and stored there. Virtual information integration refers to developing a software layer simulating an integrated database while keeping the original databases intact. The physical and virtual database integration approaches are also known as the *data warehousing* and *federated database* approaches. To evaluate queries on a federated database, one has to determine the source databases to query and to resolve data heterogeneity during query processing [15].

In the context of relational databases, the above challenges have been studied for almost three decades [24] and many interesting information integration techniques have been developed. Nevertheless, not all integration issues have be fully addressed so far as there are often hidden semantics in the data heterogeneity that are not made available to the information integration techniques that depend on them. Meanwhile, the emergence of graph data will pose additional research issues to be considered in designing information integration techniques.

This chapter will thus give an overview of information integration for graph data. Although there are applications that require different types of integrated graph data, the same integration framework and techniques can be applied. In Section 10.2, we introduce a general integration framework for graph integration before describing the techniques appropriate for the different framework components. We specifically describe a few entity resolution approaches in Section 10.3. We will highlight two graph information integration applications that have been reported in the research literature in Section 10.4. and 10.5 concludes the chapter.

10.2 Framework for Graph Information Integration

Figure 10.1 depicts the framework of integrating multiple graphs together although it is also possible to integrate two graphs at a time. The framework broadly divides the steps into *schema-level* and *instance-level integration*. The former addresses the heterogeneity of schemas of the original graphs by automatically or semi-automatically deriving the mappings between the schema elements. This is

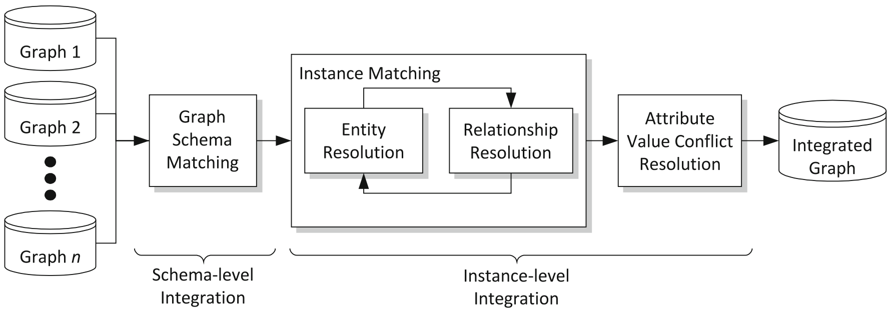


Fig. 10.1 Graph information integration framework

also known as *schema matching*. Instance-level integration refers to resolving the heterogeneity among data instances. The sub-steps include *entity resolution* and *relationship resolution* which can be carried out either sequentially or in an iterative manner. The entity resolution sub-step attempts to match graph nodes representing the same entities. The relationship resolution on the other hand matches graph edges representing the same relationships. After entity and relationship resolution, one can conduct *attribute value resolution* on the matched nodes and relationships to resolve any differences in their attributes.

The complexity involved in each integration step is correlated with the heterogeneity to be resolved. For example, in cases when the graphs to be integrated share a common schema, the graph schema matching step becomes trivial. The step can, however, be complex when (a) the underlying data models are different, (b) schemas have very different elements (i.e., node types, edge types, node attributes, and edge attributes) causing the mappings between them may not be 1-1, and (c) different constraints on the node and edge types are specified for different schemas.

Graph schema matching, depending on the data model used to represent graph instances, can be quite similar to schema matching involving relational data instances [22]. Many techniques already developed for relational schema matching are still applicable. Attribute value resolution is a step conducted on matched entities or matched relationships and is largely not affected by the graph structures. Again, the existing attribute value resolution techniques for relational data can be employed. Between entity and relationship resolutions, entity resolution has been intensively studied while relationship resolution is still a relatively less studied problem because most of the graphs to be integrated have relationships uniquely identified by their associated entities. This makes relationship resolution relatively straightforward. In other words, if entities A and B are linked with a relationship in each of two given graphs, the two relationships are identified to be the same. In this chapter, we shall therefore focus largely on the entity resolution step and its associated approaches.

Entity resolution for graph data is quite different from that for relational data due to the existence of links in graphs. When two nodes are to be determined as the same entity, one has to consider the links and even link structures connected to the two entities. We would ideally like the two nodes to share the same links or link structures if they are the same entity. This is, however, impossible unless some of the neighboring nodes have been resolved a prior. A naive approach to break this chicken and egg situation is to ignore links and simply apply entity resolution techniques for relational data. Such an approach is non-ideal and we thus introduce other techniques that consider links.

10.3 Entity Resolution for Graphs

Entity resolution, sometimes also known as *record linkage* and *entity de-duplication*, for graphs essentially identifies nodes that model same real-world entities. When two nodes model the same real-world entity, they form a *matched entity pair*. There have been many approaches proposed to determine matched entity pairs. Some of them assume that no two nodes in the same graph model the same real-world entity. In a more general integration setting, this assumption may not hold and one may have to perform entity resolution on a single graph.

The general strategy of entity resolution is depicted by the following steps:

1. Compute similarities for entity pairs.
2. User(s) judge if the pairs are matched.

The first step heavily depends on the definition of *inter-entity similarity*. Given two or more graphs, we can measure the inter-entity similarity using a similarity function $sim(e_i, e_j)$. When each graph has N entities, the number of entity pairs for similarity computation will be N^2 in the worst case. To reduce the number of entity pairs in steps 1 and 2, a *minimum similarity threshold* can be introduced. Omar et al., in the context of relational database integration exploited properties of the matching and merging entities to reduce the computation and judgment overheads of entity resolution [1].

Bhattacharya and Getoor provided a comprehensive survey of entity resolution approaches in [3]. In the following, we adopt their classification of entity resolution approaches and briefly describe them. Entity resolution can be solved in both supervised or non-supervised approaches. The former requires training data while the latter does not. In the research literature, there is a variety of supervised entity resolution approaches [8, 23] but due to space constraint, we will only cover the non-supervised ones below.

10.3.1 Attribute-Based Entity Resolution

Attribute-based entity resolution compares the attribute values of entities in order to match them. It has been widely used in resolving entity identities in relational databases and graphs due to its simplicity. Attributes or combinations of attributes that can identify entities either strongly or weakly are used in attribute-based entity resolution approaches. Attributes or attribute combinations that strongly identify entities are the entity identifiers, e.g., social security number (or SSN), person name and birthdate, and mobile number. The other attributes or attribute combinations that increase the odds of identifying entities are then known to identify entities weakly, e.g., birth year, land phone number, home address, and company address. According to the nature of the attributes used in entity resolution, there can be a few different ways of defining entity level similarity function $sim(e_i, e_j)$.

The $sim_{attrib}(e_i, e_j)$ can be defined to be either 0 or 1 only depending on the outcome of comparing the entity attributes $e_i.attrib$ and $e_j.attrib$. For example, the following inter-entity similarity function $sim_{\mathsf{SSN}}(e_i, e_j)$ returns a binary value whenever e_i and e_j have identical social security number. The second function $sim_{\mathsf{homeaddress,birthyear}}(e_i, e_j)$ returns *Jaccard* similarity metric value of their home address, a value between 0 and 1, indicating how likely e_i and e_j are matched entities when their birth year values are the same, and 0 when the birth year values are different:

$$sim_{\mathsf{SSN}}(e_i, e_j) = \begin{cases} 1 & \text{if } e_i.\mathsf{SSN} = e_j.\mathsf{SSN} \\ 0 & \text{otherwise,} \end{cases}$$

$$sim_{\mathsf{home\ address,birth\ year}}(e_i, e_j) = \begin{cases} Jaccard(e_i.\mathsf{home\ address}, & \text{if } e_i.\mathsf{birth\ year} = \\ \qquad e_j.\mathsf{home\ address}) & \quad e_j.\mathsf{birth\ year} \\ 0 & \text{otherwise.} \end{cases}$$

The *Jaccard* similarity metric of two strings of word tokens s_i and s_j is defined by

$$Jaccard(s_i, s_j) = \frac{\text{number of common words between } s_i \text{ and } s_j}{\text{number of distinct words in } s_i \text{ and } s_j}.$$

A survey of similarity metrics for string attributes such as *Jaccard* can be found in [7]. Similarity metrics for numeric attributes include normalized difference, cosine similarity, and euclidean distance [14]. Given that entities usually have multiple attributes, these similarity metrics can be combined in various ways to determine entity similarity.

10.3.2 Relational Entity Resolution

Relational entity resolution essentially involves the use of link connectivity of entities to determine how similar the entities are. Bhattacharya and Getoor proposed several relational entity resolution approaches [3] to improve over using direct attribute entity resolution.

In the *naïve relational entity resolution* approach [3], the inter-entity similarity is derived by applying an attribute-based similarity function $sim_A(e_i, e_j)$ and another edge-based similarity $sim_H(e_i, e_j)$ on the pair of entities to be matched. The overall inter-entity similarity is defined by

$$sim(e_i, e_j) = (1 - \alpha) \cdot sim_A(e_i, e_j) + \alpha \cdot sim_H(e_i, e_j),$$

where $0 \le \alpha \le 1$. The function $sim_H(e_i, e_j)$ is determined by the similarity of edges of e_i and e_j and can be defined as the aggregated similarity between the edges of e_i and e_j (denoted by $e_i.edges$ and $e_j.edges$ respectively), i.e.,

$$sim_H(e_i, e_j) = \sum_{l_i \in e_i.edges, l_j \in e_j.edges} sim_H(l_i, l_j).$$

The similarity between a pair of edges can then be defined by

$$sim_H(l_i, l_j) = Max_{e_s \in E_i, e_t \in E_j} sim_A(e_s, e_t),$$

where E_i and E_j denote the entities linked to l_i and l_j, respectively.

In the *Simrank* approach proposed by Jeh and Widom [12], the inter-entity similarity is defined by a random walk process on neighbors of entity pairs. Suppose a graph is directed and I_i (and I_j) denotes the set of in-neighbors of e_i (and e_j), the Simrank similarity between entities e_i and e_j is defined by

$$sim_{simrank}(e_i, e_j) = \begin{cases} 1 & \text{if } e_i = e_j \\ \frac{C}{|I_i||I_j|} \sum_{e'_i \in I_i} \sum_{e'_i \in I_i} sim_{simrank}\left(e'_i, e'_j\right) & \text{otherwise,} \end{cases}$$

where C is a decay factor constant between 0 and 1. Unlike the earlier approach, Simrank does not use attribute-based similarity function at all. Simrank also requires the graph to be directed. For it to work on undirected graph, a simple way is to replace an undirected edge by two directed edges, one for each direction.

10.3.3 Collective Relational Entity Resolution

The main idea of collective relational entity resolution is to group entities into *entity clusters* each representing a group of entities that model a real-world entity. The inter-entity similarity is thus measured by a combination of how similar a pair of entities e_i and e_j are by their attributes, and how similar they are by the cluster labels of their neighbors. Bhattacharya and Getoor proposed an agglomerative clustering algorithm that group entity clusters into larger entity clusters incrementally using the inter-entity cluster similarity function [3]:

$$sim(c_i, c_j) = (1 - \alpha) \cdot sim_A(c_i, c_j) + \alpha \cdot sim_R(c_i, c_j),$$

where $sim_A(c_i, c_j)$ and $sim_R(c_i, c_j)$ represent the attribute-based and relational similarities between two entity clusters c_i and c_j, respectively. The former can be determined by an aggregated attribute-based similarity between entities from c_i and c_j. The latter, $sim_R(c_i, c_j)$, is determined by the number of common cluster labels among the neighbors of entities in c_i and c_j. This neighborhood similarity can be measured using a variety of functions including *common neighbors*, *jaccard similarity*.

Instead of agglomerative clustering, Bhattacharya and Getoor also introduced a Latent Dirichlet Allocation (LDA) model for conducting collective relational entity resolution for authors of a set of publications using probabilistic model [2]. Here, author entity clusters are represented by latent authors. The observed author entities are assumed to be generated by these latent authors. Learning the mapping from observed author entities to latent authors is thus a problem of learning LDA model parameters.

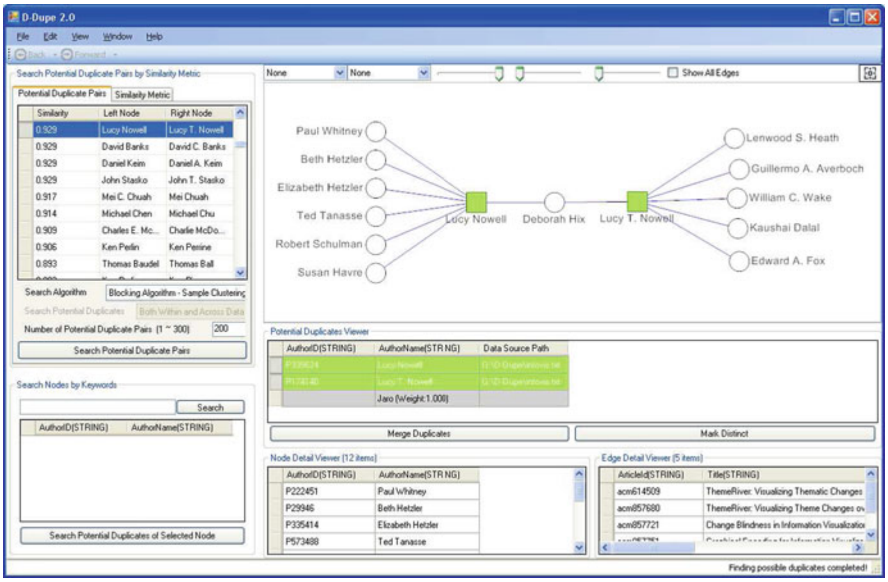
10.4 Example Applications

In this section, we shall describe two example applications of entity resolution to social network analysis. The first is *D-Dupe*, an interactive tool for entity resolution [4]. The second is *SSnetViz*, an application to visualize and explore integrated heterogeneous social networks [19].

10.4.1 D-Dupe

D-Dupe is an interactive application specially designed to resolve entity identities for authors of publications [4]. The graphs to be integrated have authors as nodes and co-authorships as edges. Each edge is thus associated with a set of publications between the connected authors. Figure 10.2 shows the user interface design of D-Dupe using the Infoviz data set that comes with the application demo [9]. It consists of mainly three user interface components including

– *Search Panel*: The search panel shows a list of entity pairs ranked by their inter-entity similarity values. Different attribute and relational similarity metrics can be chosen by the user. The panel also supports string search on author entities so as to find potential duplicates of the selected authors.
– *Graph Visualizer*: The graph visualizer displays a subgraph with a selected candidate pair of duplicate authors (e.g., “Lucy Nowell” and “Lucy T. Nowell” in the figure), their common co-authors in-between (“Deborah Hix” in the figure) and other non-common co-authors at the sides. The purpose of graph visualizer is to allow user to easily tell whether the candidate duplicate authors are indeed the same author. The user can then decide whether to merge the two candidate authors and to mark them distinct.



Similarity	Left Node	Right Node
0.929	Lucy Nowell	Lucy T. Nowell
0.929	David Banks	David C. Banks
0.929	Daniel Keim	Daniel A. Keim
0.929	John Stasko	John T. Stasko
0.917	Mei C. Chuah	Mei Chuah
0.914	Michael Chen	Michael Chu
0.909	Charles E. Mc...	Charlie McDo...
0.906	Ken Perlin	Ken Perrine
0.893	Thomas Baudel	Thomas Ball

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

AuthorID(STRING)	AuthorName(STRING)

AuthorID(STRING)	AuthorName(STRING)	Data Source Path
[illegible]	Lucy Nowell	[illegible]
[illegible]	Lucy T. Nowell	[illegible]
	Jaro (Weight:1.000)	

AuthorID(STRING)	AuthorName(STRING)
P222451	Paul Whitney
P29946	Beth Hetzler
P335414	Elizabeth Hetzler
P573488	Ted Tanasse

ArticleId(STRING)	Title(STRING)
acm614509	ThemeRiver: Visualizing Thematic Changes
acm857680	ThemeRiver: Visualizing Theme Changes ov
acm857721	Change Blindness in Information Visualizatio

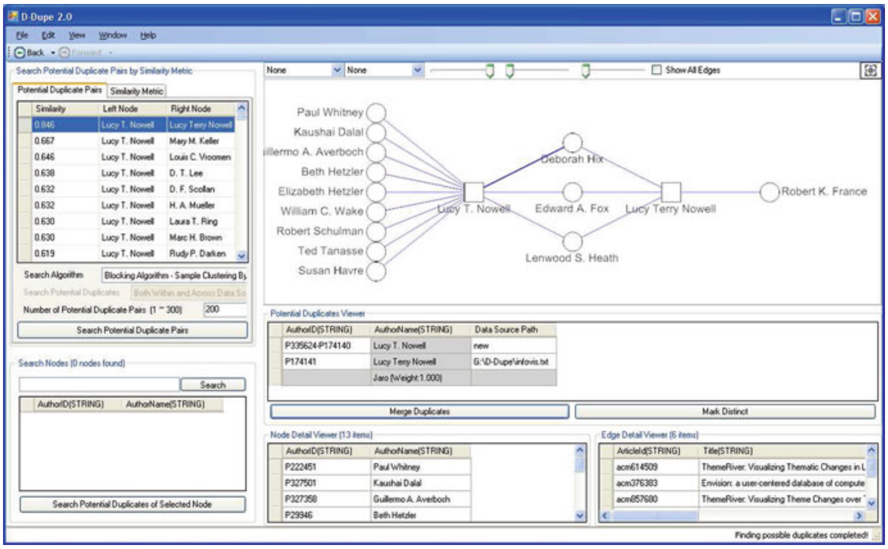
Fig. 10.2 User interface of D-Dupe

– *Detail Viewer*: The detail viewer provides detailed views of selected author candidates, their co-authors' nodes and edges in the graph visualizer.

The layout of the author graph is such that there is always some node(s) between the selected author candidates as it is not possible to have both of them appearing in the same publication. The size of an author node can be based on the author's number of publications. Both nodes and edges can be filtered for clearer viewing. Once the user selects two candidate authors to be merged together, the graph visualizer will show the two nodes combined into one and their two sets of edges are assigned to the new node labeled by "Lucy T. Nowell" as shown in Fig. 10.3. D-Dupe automatically selects the next node most similar to the merged node, i.e., "Lucy Terry Nowell," as a candidate entity to be merged. The entity resolution steps can repeat till all author entities are appropriately merged.

10.4.2 SSnetViz

SSnetViz, unlike D-Dupe, is an ongoing research project to explore and analyze heterogeneous social networks integrated from multiple data sources [19]. A heterogeneous social network is one with multiple node types (or entity types) and link types (or relationship types). In SSnetViz, nodes of the same type share the same attribute set while links are not assigned any attributes. SSnetViz provides some functions to integrate multiple heterogeneous social networks and data exploration features that allow the data sources to be identified even after entities are merged.


D-Dupe 2.0
File Edit View Window Help
Back Forward
Search Potential Duplicate Pairs by Similarity Metric
Potential Duplicate Pairs
Similarity Metric
Search Algorithm
Blocking Algorithm - Sample Clustering By
Search Potential Duplicates
Both Within and Across Data So
Search Potential Duplicate Pairs
Search Nodes (0 nodes found)
Search
Search Potential Duplicates of Selected Node
None
Show All Edges
Paul Whitney
Kaushai Dalal
illermo A. Averboch
Beth Hetzler
Elizabeth Hetzler
William C. Wake
Robert Schulman
Ted Tanasse
Susan Havre
Lucy T. Nowell
Deborah Hix
Edward A. Fox
Lenwood S. Heath
Lucy Terry Nowell
Robert K. France
Potential Duplicates Viewer
Merge Duplicates
Mark Distinct
Node Detail Viewer (13 items)
Edge Detail Viewer (6 items)
Finding possible duplicates completed!

Similarity	Left Node	Right Node
0.846	Lucy T. Nowell	Lucy Terry Nowell
0.667	Lucy T. Nowell	Mary M. Keller
0.646	Lucy T. Nowell	Louis C. Vroomen
0.638	Lucy T. Nowell	D. T. Lee
0.632	Lucy T. Nowell	D. F. Scollan
0.632	Lucy T. Nowell	H. A. Mueller
0.630	Lucy T. Nowell	Laura T. Ring
0.630	Lucy T. Nowell	Marc H. Brown
0.619	Lucy T. Nowell	Rudy P. Darken

Number of Potential Duplicate Pairs (1 ~ 300) 200

AuthorID(STRING)	AuthorName(STRING)

AuthorID(STRING)	AuthorName(STRING)	Data Source Path
P335624-P174140	Lucy T. Nowell	new
P174141	Lucy Terry Nowell	G:\D-Dupe\infovis.txt
	Jaro (Weight:1.000)	

AuthorID(STRING)	AuthorName(STRING)
P222451	Paul Whitney
P327501	Kaushai Dalal
P327358	Guillermo A. Averboch
P29946	Beth Hetzler

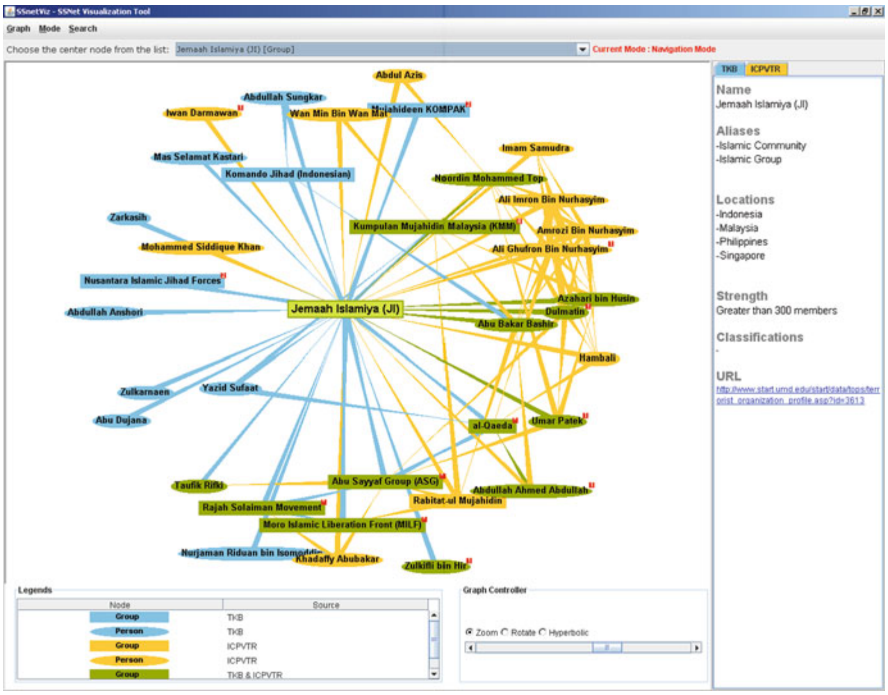
ArticleId(STRING)	Title(STRING)
acm614509	ThemeRiver: Visualizing Thematic Changes in L
acm376383	Envision: a user-centered database of compute
acm857680	ThemeRiver: Visualizing Theme Changes over "

Fig. 10.3 After merging "Lucy Nowell" and "Lucy T. Nowell"

The design of SSnetViz is originally motivated by the need to support data analysis of terrorism related social networks from disparate sources. The networks SSnetViz has so far integrated come from three data sources: (a) TKBNetwork: a terrorism network from the MIPT Terrorism Knowledge Base[4] (TKB); (b) PVTR-Network: a terrorism network created by the International Center for Political Violence and Terrorism Research (ICPVTR), a center focusing on gathering and analysis terrorism data using public domain data; and (c) WikiTerrorism: a network we constructed using terrorism-related articles from Wikipedia. Both TKBNetwork and PVTRNetwork are constructed by experts while WikiTerrorism represents knowledge collaboratively edited by the online community.

Figure 10.4 depicts the user interface design of SSnetViz. It consists of a *network viewer* that displays the entire social network graph or selected subgraph. Nodes of different types are shown using different shapes while the different data sources are shown using different colors. The legend of shapes and colors of nodes and links is shown at the bottom of network viewer. For example, terrorists are shown as oval nodes and terrorist groups are shown as rectangle nodes. Light blue and yellow indicate the information from TKBNetwork and PVTRNetwork respectively. When a node or link is derived by integrating two or more original networks (e.g., "Jemaah Islamiya (JI)" node in Fig. 10.4), we assign them a distinct color representing the overlapping data sources. The color and shape schemes can be configured by users for easy viewing. Zooming, rotation, hyperbolic view, node expansion, node/link

[4] MIPT is the acronym of Memorial Institute for the Prevention of Terrorism.



Choose the center node from the list: Jemaah Islamiya (JI) [Group]

TKB ICPVTR

Name
Jemaah Islamiya (JI)

Aliases
-Islamic Community
-Islamic Group

Locations
-Indonesia
-Malaysia
-Philippines
-Singapore

Strength
Greater than 300 members

Classifications
-

URL
http://www.start.umd.edu/start/data/tops/terrorist_organization_profile.asp?id=3613

Node	Source
🟦 Group	TKB
🔵 Person	TKB
🟨 Group	ICPVTR
🟡 Person	ICPVTR
🟩 Group	TKB & ICPVTR

[✓ Zoom] Rotate Hyperbolic

Fig. 10.4 User interface design of SSnetViz

hiding, node search, path search, and other operations are also provided to manipulate and explore the social network in network viewer. Whenever a node is selected, its attribute values divided according to the data source(s) the node is found will be shown in the node information panel on the right side. In Fig. 10.4, "Jemaah Islamiya (JI)" node's attribute values from TKBNetwork and PVTRNetwork are shown.

Since SSnetViz has to integrate social networks with heterogeneous schemas and node/link instances, both schema-level integration and instance-level integration are required. The integration steps depicted in Fig. 10.5 have been adopted. SSnetViz has a generic graph schema that is capable of storing and updating social networks with heterogeneous schemas. When two social networks are to be integrated, the user first matches the node types of the networks using a *node type matching module*. The step is performed once only and the matched node types are stored for subsequent instance-level integration.

Instance-level integration in SSnetViz involves mainly node matching, i.e., entity resolution. Link instances are automatically integrated if their link types are identical. Both rule based and manual node matching approaches are supported by a *node merging module* shown in Fig. 10.8.

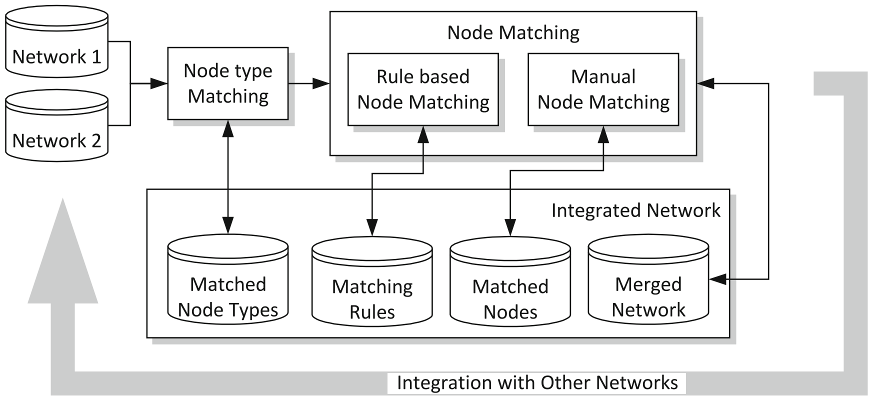


Fig. 10.5 SSnetViz's integration steps

For illustration, we show two social network subgraphs showing the one-hop neighborhoods of "Jemaah Islamiyah (JI)" from TKBNetwork and PVTRNetwork in Figs. 10.6 and 10.7 respectively. Note that the TKBNetwork and PVTRNetwork have several discrepancies in naming their entities. We would like to match their

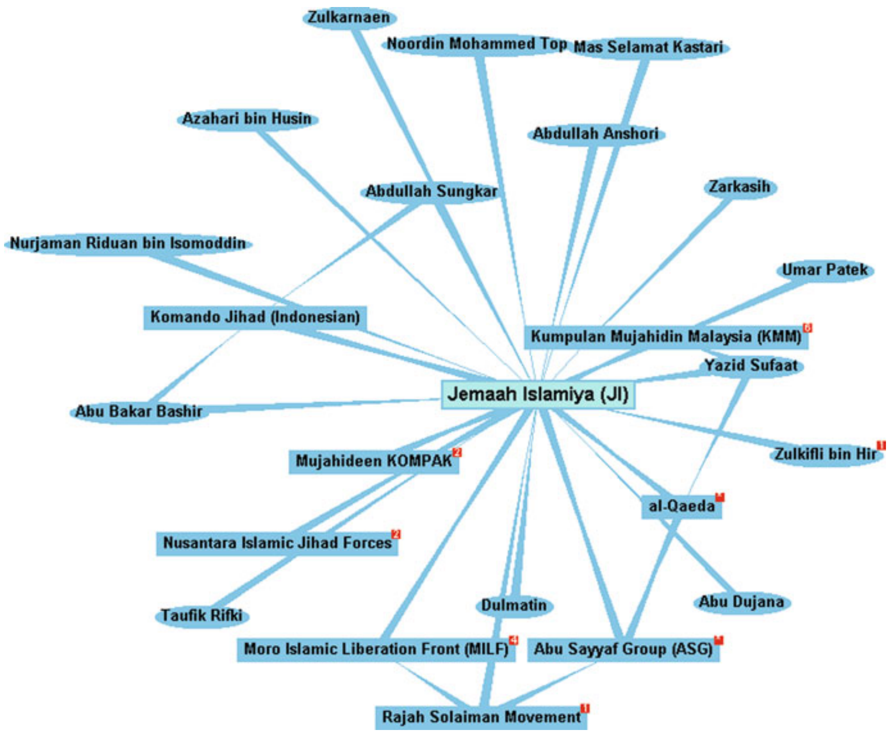


Fig. 10.6 Social network subgraph from TKBNetwork

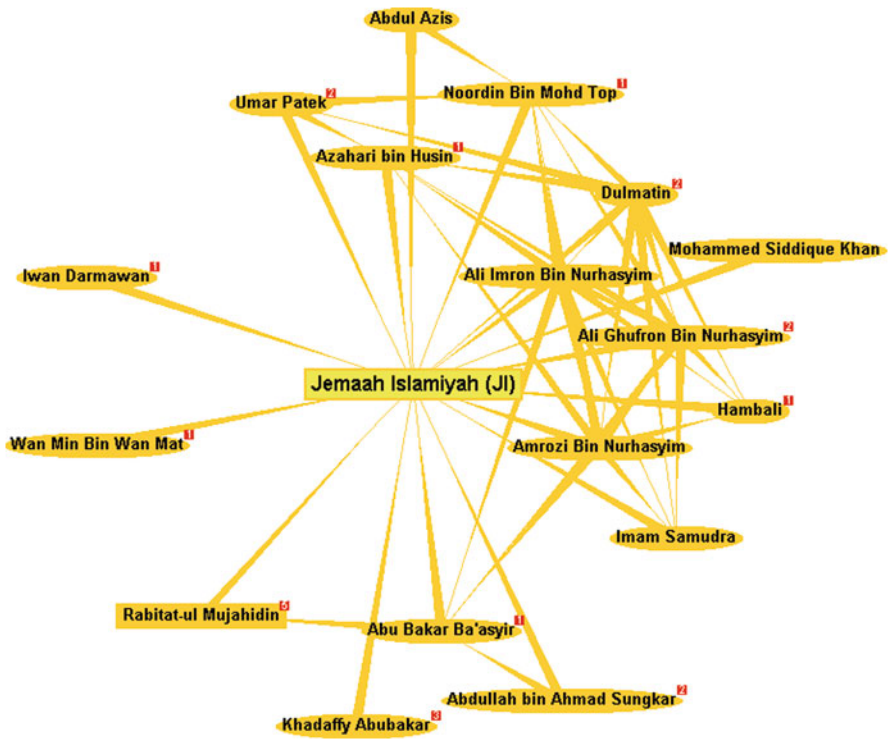


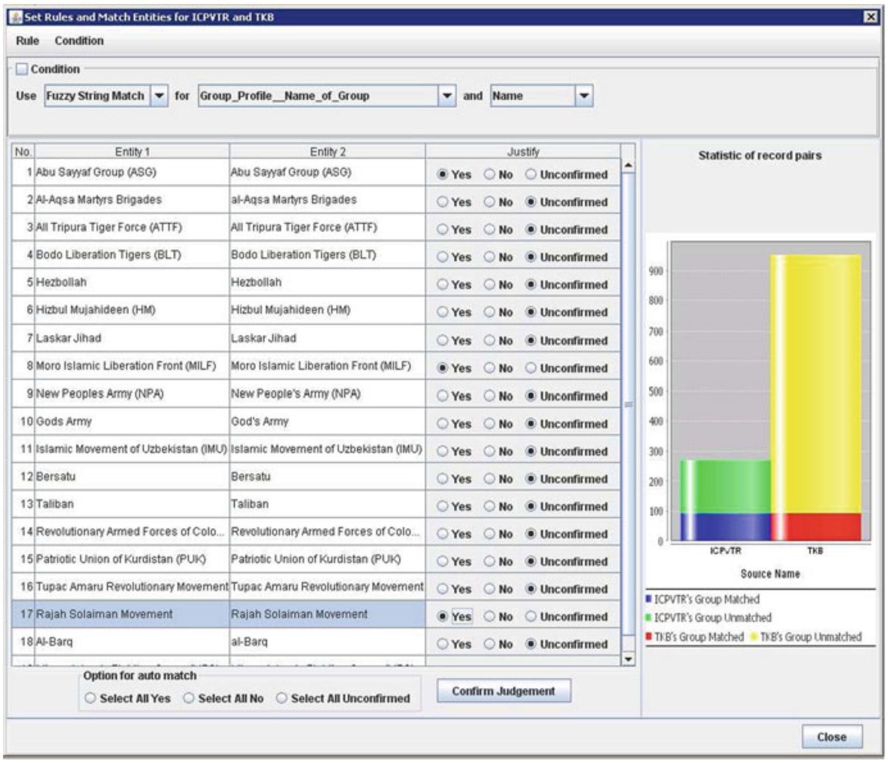
Fig. 10.7 Social network subgraph from PVTRNetwork

nodes and derive the integrated network. Table 10.1 shows a subset of matched nodes that need to be identified. Note that some of these entities have identical names in TKBNetwork and PVTRNetwork but others have some name differences, e.g., "Jemaah Islamiya (JI)" and "Jemaah Islamiyah (JI)."

In general, node instances modeling the same real-world entities may have different attribute values. SSnetViz users can match them using user-defined matching rules which specify the attribute similarity functions for generating candidate matched node pairs (see Fig. 10.8). Among the candidate matched node pairs, the user can manually merge the correct node pairs and mark the rest as incorrect ("no"

Table 10.1 Matched node pairs

	TKBNetwork	PVTRNetwork
1	Jemaah Islamiya (JI)	Jemaah Islamiyah (JI)
2	Umar Patek	Umar Patek
3	Noordin Mohammed Top	Noordin Bin Mohd Top
4	Azahari bin Husin	Azahari bin Husin
5	Dulmatin	Dulmatin
6	Abu Bakar Bashir	Abu Bakar Ba'asyir



☐ Condition

Use Fuzzy String Match for Group_Profile__Name_of_Group and Name

No.	Entity 1	Entity 2	Justify
1	Abu Sayyaf Group (ASG)	Abu Sayyaf Group (ASG)	[✓ Yes] No Unconfirmed
2	Al-Aqsa Martyrs Brigades	al-Aqsa Martyrs Brigades	Yes No [✓ Unconfirmed]
3	All Tripura Tiger Force (ATTF)	All Tripura Tiger Force (ATTF)	Yes No [✓ Unconfirmed]
4	Bodo Liberation Tigers (BLT)	Bodo Liberation Tigers (BLT)	Yes No [✓ Unconfirmed]
5	Hezbollah	Hezbollah	Yes No [✓ Unconfirmed]
6	Hizbul Mujahideen (HM)	Hizbul Mujahideen (HM)	Yes No [✓ Unconfirmed]
7	Laskar Jihad	Laskar Jihad	Yes No [✓ Unconfirmed]
8	Moro Islamic Liberation Front (MILF)	Moro Islamic Liberation Front (MILF)	[✓ Yes] No Unconfirmed
9	New Peoples Army (NPA)	New People's Army (NPA)	Yes No [✓ Unconfirmed]
10	Gods Army	God's Army	Yes No [✓ Unconfirmed]
11	Islamic Movement of Uzbekistan (IMU)	Islamic Movement of Uzbekistan (IMU)	Yes No [✓ Unconfirmed]
12	Bersatu	Bersatu	Yes No [✓ Unconfirmed]
13	Taliban	Taliban	Yes No [✓ Unconfirmed]
14	Revolutionary Armed Forces of Colo...	Revolutionary Armed Forces of Colo...	Yes No [✓ Unconfirmed]
15	Patriotic Union of Kurdistan (PUK)	Patriotic Union of Kurdistan (PUK)	Yes No [✓ Unconfirmed]
16	Tupac Amaru Revolutionary Movement	Tupac Amaru Revolutionary Movement	Yes No [✓ Unconfirmed]
17	Rajah Solaiman Movement	Rajah Solaiman Movement	[✓ Yes] No Unconfirmed
18	Al-Barq	al-Barq	Yes No [✓ Unconfirmed]

Option for auto match

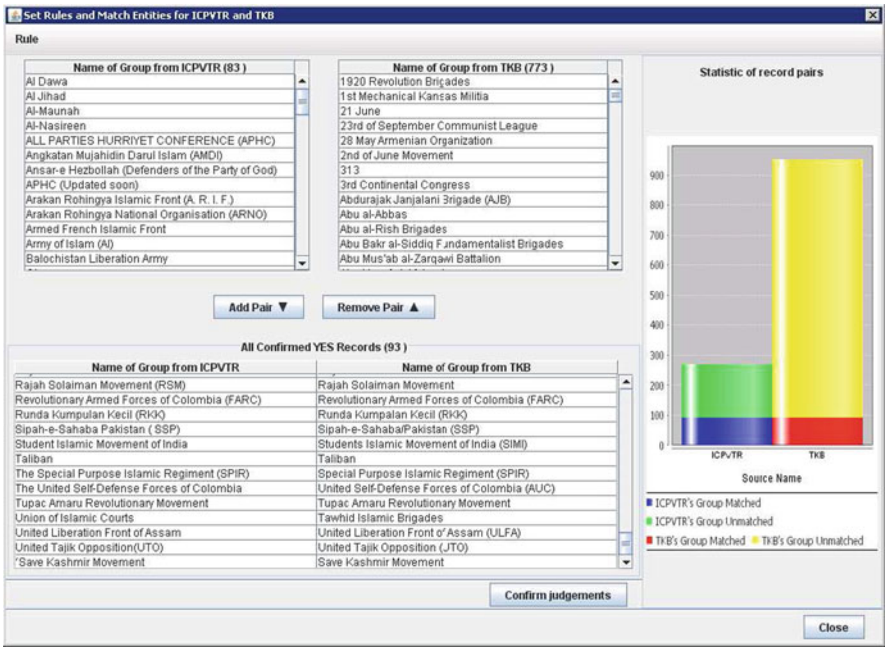
Select All Yes Select All No Select All Unconfirmed

Fig. 10.8 SSnetViz's node merging module

option in the figure) or unconfirmed. A cylindrical bar showing the numbers of matched and unmatched node instances in each social network is also shown by the node merging module. In cases where matching rules fail to include a matched node pair, a manual node merging module can be used where matched node pairs can be identified manually (see Fig. 10.9). Using these two matching approaches, the integrated network can be derived. Fig. 10.10 shows the one-hop neighborhood of "Jemaah Islamiyah (JI)" in the integrated network.

10.5 Conclusion

Graph information integration is an important class of data integration problem as graph data can be found in many databases and integrated graphs are extremely useful for complex queries, data analysis, and data mining. This chapter aims to give an overview of the integration framework. We focus on entity resolution in


Set Rules and Match Entities for ICPVTR and TKB

Rule

Name of Group from ICPVTR (83)
Al Dawa
Al Jihad
Al-Maunah
Al-Nasireen
ALL PARTIES HURRIYET CONFERENCE (APHC)
Angkatan Mujahidin Darul Islam (AMDI)
Ansar-e Hezbollah (Defenders of the Party of God)
APHC (Updated soon)
Arakan Rohingya Islamic Front (A. R. I. F.)
Arakan Rohingya National Organisation (ARNO)
Armed French Islamic Front
Army of Islam (AI)
Balochistan Liberation Army

Name of Group from TKB (773)
1920 Revolution Brigades
1st Mechanical Kansas Militia
21 June
23rd of September Communist League
28 May Armenian Organization
2nd of June Movement
313
3rd Continental Congress
Abdurajak Janjalani Brigade (AJB)
Abu al-Abbas
Abu al-Rish Brigades
Abu Bakr al-Siddiq Fundamentalist Brigades
Abu Mus'ab al-Zarqawi Battalion

All Confirmed YES Records (93)

Name of Group from ICPVTR	Name of Group from TKB
Rajah Solaiman Movement (RSM)	Rajah Solaiman Movement
Revolutionary Armed Forces of Colombia (FARC)	Revolutionary Armed Forces of Colombia (FARC)
Runda Kumpulan Kecil (RKK)	Runda Kumpalan Kecil (RKK)
Sipah-e-Sahaba Pakistan (SSP)	Sipah-e-Sahaba/Pakistan (SSP)
Student Islamic Movement of India	Students Islamic Movement of India (SIMI)
Taliban	Taliban
The Special Purpose Islamic Regiment (SPIR)	Special Purpose Islamic Regiment (SPIR)
The United Self-Defense Forces of Colombia	United Self-Defense Forces of Colombia (AUC)
Tupac Amaru Revolutionary Movement	Tupac Amaru Revolutionary Movement
Union of Islamic Courts	Tawhid Islamic Brigades
United Liberation Front of Assam	United Liberation Front of Assam (ULFA)
United Tajik Opposition(UTO)	United Tajik Opposition (JTO)
'Save Kashmir Movement	Save Kashmir Movement

Fig. 10.9 SSnetViz's manual node merging module

graphs which has made significant progress in recent years. We also introduce two example applications, D-Dupe that is specifically designed for performing interactive entity resolution on graphs, and SSnetViz that integrates and manages integrated heterogeneous social networks.

Looking ahead, there are still many interesting problems to be addressed in graph information integration. Most of the research today has largely focused on specific problems, e.g., entity resolution, without considering the other related integration problems, e.g., schema integration and attribute value conflict resolution. Designing a complete suite of solutions for all the integration problems is challenging but is certainly the direction to pursue as the cost of data integration will only increase with more graph data to be generated in the future.

At present, researchers in graph information integration are also struggling with the evaluation of different graph information integration methods. This is mainly caused by a lack of publicly available graph data sets and methodology for performance comparison. DBLP is a good example of graph data that is publicly available. The ground truths of its integration with other author–paper graphs are, however, not available. Hence, it is difficult to compare accuracies of different integration methods on the data set.

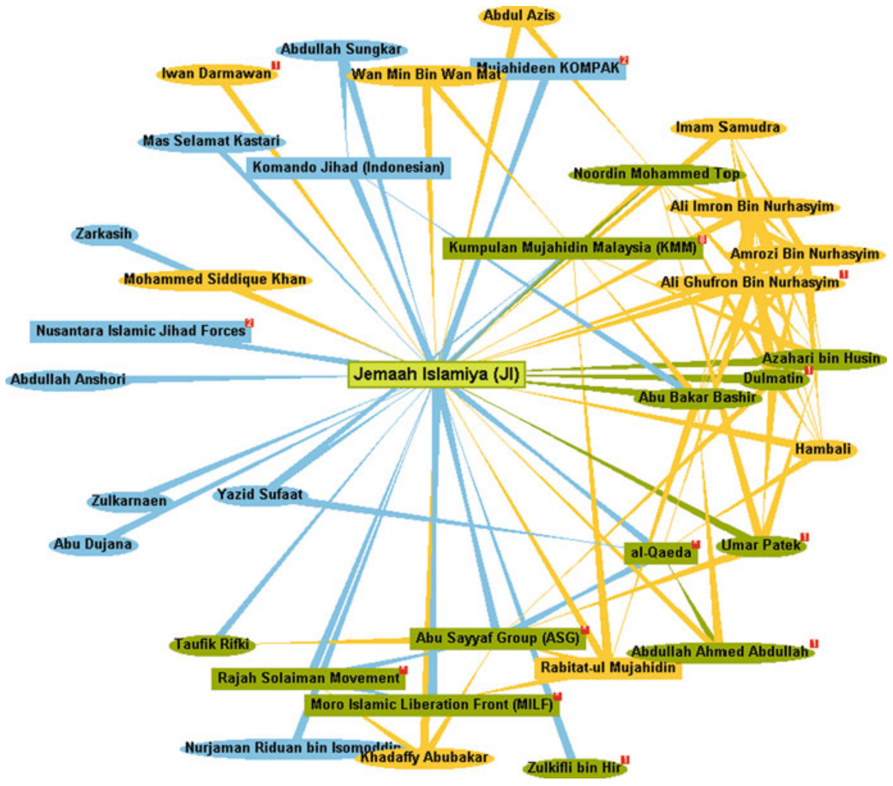


Fig. 10.10 Integrated social network subgraph

Acknowledgments We would like to acknowledge the support by A*STAR Public Sector R&D, Singapore, Project Number 062 101 0031 in the SSNet Project. We also thank Maureen and Nelman Lubis Ibrahim for implementing the SSnetViz system.

References

1. O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: A generic approach to entity resolution. *VLDB Journal*, 18(1):255–276, 2009.
2. I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *SIAM Conference on Data Mining*, Bethesda, Maryland, USA, 2006.
3. I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.
4. M. Bilgic, L. Licamele, L. Getoor, and B. Shneiderman. D-dupe: An interactive tool for entity resolution in social networks. In *International Symposium on Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 505–507, September 2005.
5. P. Buneman. Semistructured data. In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Tucson, Arizona, 1997.
6. P. Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

7. W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IJCAI Workshop on Information Integration*, pages 73–78, Acapulco, Mexico, August 2003.
8. P. Domingos. Multi-relational record linkage. In *KDD-2004 Workshop on Multi-Relational Data Mining*, pages 31–48, Seattle, Washington, 2004.
9. J.-D. Fekete, G. Grinstein, and C. Plaisant. The history of infovis. In *IEEE InfoVis 2004 Contest*, www.cs.umd.edu/hcil/iv04contest, Austin, Texas, 2004.
10. M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. Strudel: a web site management system. In *ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, 1997.
11. N. Guarino. *Formal Ontology in Information Systems*, chapter Formal Ontology in Information Systems. IOS Press, Amsterdam, 1998.
12. G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 538–543, Edmonton, Alberta, Canada, 2002.
13. A. Jhingran, N. Mattos, and H. Pirahesh. Information integration: A research agenda. *IBM Systems Journal*, 41(4):555–562, 2002.
14. L. Jin, C. Li, and S. Mehrotra. Efficient record linkage in large data sets. In *International Conference on Database Systems for Advanced Applications*, Kyoto, Japan, 2003.
15. E.-P. Lim and J. Srivastava. Query optimization and processing in federated database systems. In *ACM Conference on Information and Knowledge Management*, pages 720–722, Washington D.C., 1993.
16. E.-P. Lim, J. Srivastava, S. Prabhakar, and J. Richardson. Entity identification in database integration. In *IEEE International Conference on Data Engineering*, pages 294–301, Vienna, Austria, 1993.
17. E.-P. Lim, J. Srivastava, and S. Shekhar. An evidential reasoning approach to attribute value conflict resolution in database integration. *IEEE Transactions on Knowledge and Data Engineering*, 8(5):707–723, 1996.
18. W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Survey*, 22(3):267–293, 1990.
19. Maureen, A. Sun, E.-P. Lim, A. Datta, and K. Chang. On visualizing heterogeneous semantic networks from multiple data sources. In *International Conference on Asian Digital Libraries*, pages 266–275, Bali, Indonesia, 2008.
20. J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3), 1997.
21. A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Survey*, 22(3):183–236, 1990.
22. S. Spaccapietra and C. Parent. View integration: A step forward in solving structural conflicts. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):258–274, 1994.
23. P. Treeratpituk and C. L. Giles. Disambiguating authors in academic publications using random forests. In *Joint Conference in Digital Libraries*, Austin, Texas, June 2009.
24. P. Ziegler and K. R. Dittrich. Three decades of data integration — all problems solved? In *18th IFIP World Computer Congress (WCC 2004)*, pages 3–12, Toulouse, France, 2004.