Handling Inconsistency for Multi-Source Integration

Sheila Tejada Craig A. Knoblock Steven Minton
University of Southern California/ISI
4676 Admiralty Way, Marina del Rey, California 90292
{tejada,knoblock,minton}@isi.edu,
(310) 822-1511 x799

Abstract

The overwhelming amount of information sources now available through the internet has increased the need to combine or integrate the data retrieved from these sources in an intelligent and efficient manner. A desirable approach for information integration would be to have a single interface, like the SIMS information broker [1], which allows access to multiple information sources. An example application is to retrieve all the menus of restaurants from Joe's Favorite Restaurants site which have been rated highly by the Department of Health.

This task, if performed manually, would require a significant amount of work for the user. An information broker, like SIMS, would allow access to multiple information sources, abstracting away the need for the user to know the location or query access methods of any particular source. SIMS stores knowledge about the data contained in each of these sources, as well as the relationships between the sources in the form of a model, called a domain model. The first step in creating the domain model is to determine which data instances appear in multiple sources, e.g. which restaurants from Joe's web site, like "Art's Deli," also appears on the Health Department's site. Once the common data instances are determined, the relationship between the data in the sources can be modeled in the domain model as subset, superset, equality, or overlapping

A special case for information integration is when data instances can exist in inconsistent formats across several sources, e.g. the restaurant "Art's Deli" can appear as "Art's Delicatessen" in another source. For the integration process each source can be seen as a relation; therefore, integrating the sources requires performing a join on the two relations by comparing the instances of the primary keys. Since the instances have inconsistent formats, some mapping information is needed to map one instance to another e.g. ("Art's Deli" from Joe's source to "Art's Delicatessen" from the Department of Health site). This information can be stored in the form of a mapping table, or as a mapping function, if a compact translation can be found to accurately convert data instances from one source into another. Once a mapping construct is created it can be modeled as a new information source. This integration technique allows SIMS to properly integrate data across several sources that contain inconsistent data instances.

Presently, mapping constructs are generated manually, but we are developing a semi-automate approach. The figure contains tuples from Joe's Restaurant source and the matching tuples from the Health Department:

Name Address Phone

1. (Art's Deli, 342 Beverly Blvd, (310)302-5319)
   (Art's Delicatessen,342 Beverly Boulevard,310-302-5319)

2. (CPK, 65 La Cienga Blvd, 310-987-8923)
   (California Pizza Kitchen,65 La Cienga Blvd,310-987-8923)

3. (The Beverly, 302 MLK Blvd, 213-643-2154)
   (Cafe Beverly, 302 Martin Luther King Jr. Boulevard,645-4278)

Figure 1: *Matched Restaurant Tuples*

The key idea behind our approach for generating mapping constructs is to compare all of the shared attributes of the sources in order to determine which tuples are matched, (**Name** with **Name**, **Address** with **Address,** and **Phone** with **Phone**). Since the data instances in the sources are represented in inconsistent formats they can not be compared using equality, but must be judged according to similarity. To determine the similarity between strings we developed general domain independent transformation rules to recognize transformations like substring, acronym, and abbreviation. For example, "Deli" is a substring transformation of "Delicatessen." A probabilistic similarity measure is calculated for each of the transformations between the strings of the two data instances; and then they each are combined to become the similarity measure of the data instances for that attribute. When comparing the data instance "Art's Deli" with "Art's Delicatessen," the probabilities are calculated for the string transformations of "Art's" to "Art's" and "Deli" to "Delicatessen." These probabilities are combined to be the similarity measure for the two instances. After the similarity measures are determined for each of the attributes, **Name**, **Address** and **Phone**, then they are combined to measure the similarity for the two tuples. The most probable matching between the two sets of tuples is then determined. We are employing a statistical learning technique in order to iteratively refine the initial probability measures to increase the accuracy of the matches. Once the mapping is known then a mapping table or function can be created using the instances from the primary key attribute.

Some related work has conducted by Huang & Russell [2] on matching tuples across relations using a probabilistic appearance model. Their approach also incorporates the idea of comparing the tuples based on of all of the shared attributes. To determine similarity between two instances, they calculate the probability that given one instance it will appear like the second instance in the other relation. Calculating these probabilities requires a training set of correctly paired tuples (like the tuples in the figure). Unfortunately, appearance probabilities will not be helpful for an attribute with a unique set of instances, like **Restaurant Name**. Since "Art's Deli" only occurs once in the set of instances, knowing that it appears like "Art's Delicatessen" does not help in matching any other instances. In our approach the initial probability measures are calculated for the strings of instances that match using a specific transformation rule; therefore, having matched "Art's Deli" with "Art's Delicatessen" will increase the probability of "Deli" as a substring of "Delicatessen." This will increase the probability for other instances which use that specific transformation.

Other related work by Cohen [3] determines the mappings by using the IR vector space model to perform similarity joins on the primary key. In this work stemming is used to measure similarity between strings; therefore, in the figure "CPK" would not match "California Pizza Kitchen." But, if the values from the other attributes were taken into consideration, the tuples would match. In experiments where an entire tuple is treated as one attribute accruracy was reduced. As illustrated by the third example, "The Beverly" from Joe's site would equally match the tuples for "Art's Delicatessen" and "Cafe Beverly." Our approach would be able to handle all of these examples, because it combines all of the measurements calculated individually for each shared attributes to determine the correct mapping.

References

[1] Arens, et.al. Query Processing in the SIMS Information Mediator. Advanced Planning Technology, editor, Austin Tate, AAAI Press, Menlo Park, CA, 1996.

[2] Timothy Huang and Stuart Russell. Object Identification in Bayesian Context. Proceedings of IJCAI-97. Nagoya, Japan 1997.

[3] William W. Cohen. Knowledge integration for structured information sources containing text. The SIGIR-97 Workshop on Networked Information Retrieval, 1997.