Knowledge Gathering and Matching in Heterogeneous Databases [†]

Wen-Syan Li
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, Illinois, 60208-3118
acura@eecs.nwu.edu

Abstract

In order to integrate a wide variety of databases or many diverse sources of information, we need the ability to learn the similarities directly from instances of the data, which may be embodied within a database model, a conceptual schema, application programs, or data contents. The process of determining semantically equivalent data items can not be "pre-programmed" since the information to be accessed is heterogeneous. Intelligent information integration involves extracting semantics, expressing them as metadata, and matching semantically equivalent data elements. Semint (SEMantic INTegrator) is a system prototype for semantic integration being developed at Northwestern University. It provides a graphical user interface and supports access to a variety of database systems and utilizes both schema information and data contents to determine attribute equivalence. In Semint, the knowledge of how to match equivalent data elements is "discovered", not "pre-programmed".

Introduction

Applications in a wide variety of industries require access to multiple heterogeneous distributed databases. One reason is that enterprises have various kinds of databases due to company merge or due to introduction of new database technology. Another reason is because of the increasing need for integration of information.

In semantic integration, attributes (classes of data items) are compared in a pairwise fashion to determine their equivalence. Identifying semantically related objects and then resolving the schematic differences is the *fundamental* question in any approach to database system interoperability.

The number of database integrated may be from few databases (can be integrated to a single, tightly integrated system) to thousands of databases (e.g. US West [Drew *et al.*, 1993] or information available on Internet). Manually comparing all possible pairs of attributes is an unreasonably large task. [Goh *et al.*, 1994] argues existing integration strategies might provide satisfactory support for small or static systems, but not for large-scale interoperable database systems operating in a dynamic environment. US West reports having 5 terabytes of data managed by 1,000 systems, with customer information alone spread across 200 different databases [Drew *et al.*, 1993]. One group at GTE began the integration process with 27,000 data elements, from just 40 of its applications. It required an average of four hours per data element to extract and document matching elements when the task was performed by someone other than the data owner [Ventrone and Heiler, 1994]. Other GTE integration efforts have found the elements overlapped or nearly matched in their database to be close to 80% [Ventrone and Heiler, 1994].

Existing techniques concentrate either on interactive support for manual semantic integration as part of automated or on comparing attribute names to automatically determine similar attributes. In the latter approach, the names of attributes are compared to determine similar attributes. Problems occur with synonyms (objects with different names that represent the same concepts); these are handled using a (manually created) "synonym table". Another problem is homonyms: Names are the same but different concepts

[†]This material is based upon work supported by the National Science Foundation under Grant No. CCR-9210704.

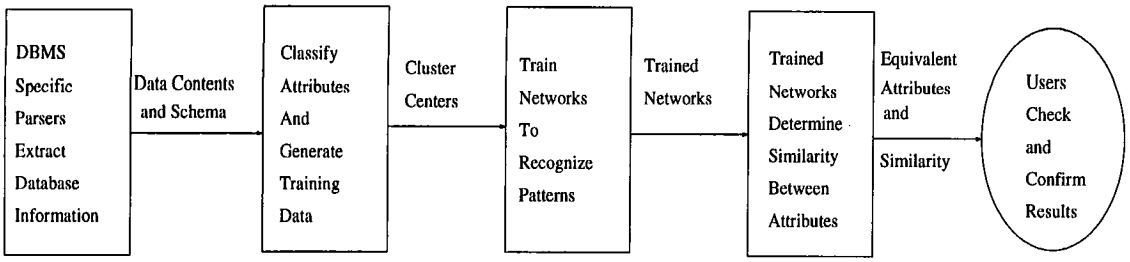

Figure 1: Overview of Semantic Integration Procedure in Semint

are represented. From GM's efforts in integration [Premerlani and Blaha, 1994], attribute names were not sufficient for semantic integration; only a few obvious matches were found. However, similarities in schema information were found to be useful. For example, it was discovered in one case that attributes of type char(14) were equivalent to those of char(15) (with an appended blank). Semint contains two novel aspects in data integration:

1. Information on the schema and data contents, rather than attribute names, is used to determine attribute similarity (note that this is compatible with name-based semantic integration techniques; a production system would probably use both).
2. The means of determining similarity (in other words, *how* the information is used to determine similarity) is *learned on a per-database basis*, rather than *pre-programmed*.

This paper will discuss how Semint extracts information from a database and learns how to determine attribute similarity directly from that information. We will first give some technical details of Semint, then discuss some possible extensions.

System Architecture

Semint [Li and Clifton, 1995] is a system prototype for semantic integration being developed at Northwestern University. It can operate in a graphical interactive mode (allowing users to provide known information about the semantic integration problem at hand), or automatically. It is implemented using C and Motif, and runs on IBM RS6000s under AIX and Sun workstations under Sun OS. Databases to be integrated are accessed directly using automatic "catalog parsers". Figure 1 outlines the semantic integration process in Semint. In this process, DBMS specific parsers extract information (schema and data content statistics) from databases and transform them into a single format (so these information can be compared). Then, a classifier is used to learns how to discriminate among attributes in a single database. The classifier output, the weights of cluster centers, is used to train a neural network to recognize categories; this network can then determine similar attributes between databases.

DBMS Specific Parser

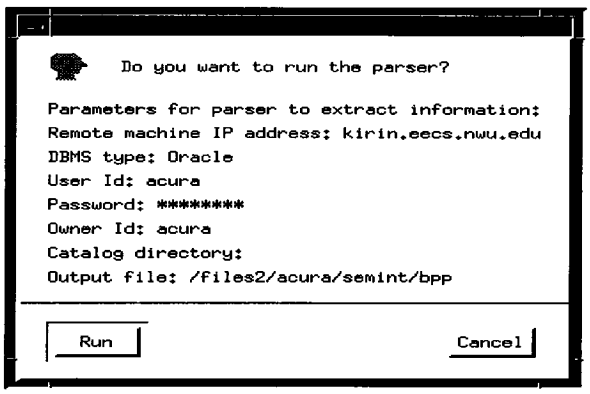

Figure 2: Parser Window in Semint

In Semint, users only need to specify the DBMS type (currently we are testing Oracle, Ingres, and "flat file" parsers; other parsers will be developed as resources allow). The user then needs to supply DBMS-specific connection information for the desired database. In Oracle7, user id, user password, and table owner id are necessary. The Oracle7 parser window in Semint is shown in Figure 2. DBMS specific parsers are implemented using SQL embedded C (e.g. Pro*C in Oracle7). Although different DBMS's use different data dictionaries to contain schema information and integrity constraints, these DBMS specific parsers are similar. Semint automatically extracts schema information and constraints from the database catalogs and

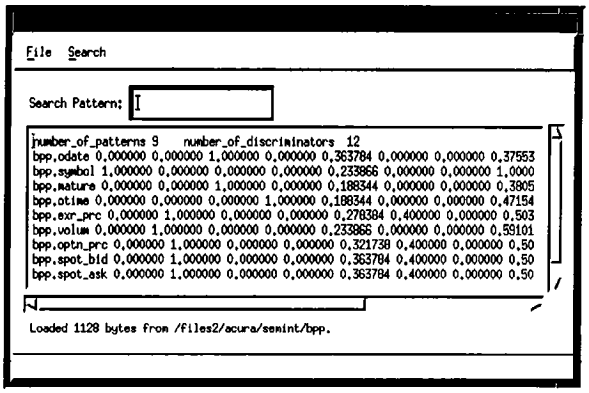

Figure 3: Parser Output

statistics on the data contents using queries over the data. It may take few hours to scan through a table with several million rows of data. However, only a small portion of sample data is needed (e.g. picked 1 row out of 10, not first 10% of the rows) to compute statistics on the data contents. The size of a table can be found in the catalog, then we decide the percentage of rows we want to sample. The information extracted from different databases is then transformed into a single format and normalized (see [Li and Clifton, 1994] for details). The parser output is shown in Figure 3.

Schema Information The schema information used by Semint includes data types, length, scale, precision, and the existence of constraints (primary keys, foreign keys, candidate keys, value and range constraints, disallowing null values, and access restrictions). The sample SQL queries (in Oracle7 Pro*C) extracting schema information and constraints from Oracle7 databases are shown in Figure 4. In some cases (such as flat-file data) we may not have an accessible schema definition. Many of the above characteristics can be determined by inspecting the data. This need not be a manual process, commercial tools are available that can automatically extract schema information from flat files.

Data Content Statistics The data contents of different attributes tend to be different even though their schema designs may be the same. This shows up in their data patterns, value distributions, grouping or other characteristics. These can serve to "characterize" attributes. For example, "SSN" and "Account balance" can all be designed as nine-digit numerical fields; they may not be distinguishable based solely on

```
SELECT a.TABLE_NAME, a.COLUMN_NAME, a.DATA_TYPE,
       a.DATA_LENGTH, a.NULLABLE, a.DATA_SCALE,
       a.DATA_PRECISION, a.DATA_DEFAULT,
       a.NUMBER_DISTINCT
  FROM USER_TAB_COLUMNS a, ALL_TAB_COLUMNS b
  WHERE a.TABLE_NAME=b.TABLE_NAME
    AND a.COLUMN_NAME=b.COLUMN_NAME
    AND b.OWNER=:owner_id;

SELECT DISTINCT CONSTRAINT_TYPE
  FROM USER_CONSTRAINTS a, USER_CONS_COLUMNS b
  WHERE a.CONSTRAINT_NAME=b.CONSTRAINT_NAME
    AND b.TABLE_NAME=:table_name;
    AND b.COLUMN_NAME=:column_name;
```

Figure 4: SQL extracting schema and constraints

```
SELECT AVG(:column_name), MAX(:column_name),
       MIN(:column_name),STDDEV(:column_name),
       VARIANCE(:column_name)
  FROM :table_name;

SELECT AVG(VSIZE(:column_name)),
       MAX(VSIZE(:column_name)),
       MIN(VSIZE(:column_name)),
       STDDEV(VSIZE(:column_name)),
       VARIANCE(VSIZE(:column_name))
  FROM :table_name;
```

Figure 5: SQL Extracting Data Content Statistics

their schema characteristics. However, their data patterns such as value distributions, and averages are all different. Thus, examining data contents can correct or enhance the accuracy of the outcomes from the dictionary level and the schema level. In [Li and Clifton, 1993] we utilize schema information to identify matching attributes. [Li and Clifton, 1994] includes statistics on data contents to determine attribute similarity. The statistics on data contents used in Semint include maximum, minimum, average (mean), variance, coefficient of variance, existence of null values, existence of decimals, scale, precision, ratio of the number of numerical characters to the total number of characters, ratio of white-space characters to total characters, grouping, and number of segments. For numerical fields, we use their values to compute statistics. For character fields, whose values are not computable, we compute statistics on numbers of bytes actually used to store data (it is meaningless to compute statistics on ASCII code numbers). The sample SQL queries extracting data content statistics from Oracle7 databases for character and numerical fields are shown in Figure 5:

Data Type Conversion The *internal* data types used in Oracle7 are Char, Long, Varchar, Number, Date, Raw, Long raw, Mlslabel (binary format of an

operating system label), and Rowid. We categorize Raw and Long raw, and Char, Long, and Varchar into one group respectively since their length can separate them from each other. We use internal data types to match fields within sample DBMS. To match fields in different DBMS's, we use external data types. Normally a DBMS has more external data types than internal data types. For example, there are 9 internal and 21 external data types in Oracle7. Oracle7 has only one internal numerical data type - Number, but it has Real, Integer, and other external numerical data types. Simply considering scale and precision of a numerical field, we can convert a Number internal data type into Real, Integer, or other external numerical data types in order to match fields in other DBMS. "Behavior semantics" such as use of cross reference, view, cluster, sequence, synonym, and dependence can also be extracted from system data dictionary. We are currently incorporating these "behavior semantics" into our Oracle7 parser.

Classifier

The available information from an individual database discussed above is used as input data for a classifier to categorize attributes within a single DBMS. Semint uses the self-organizing map algorithm, an *unsupervised* learning algorithm, as the classifier. We have adapted this algorithm so that users can determining how fine these categories are by setting the radius of clusters (threshold). A window dump of classifier is shown in Figure 6. The classifier output is shown in Figure 7.

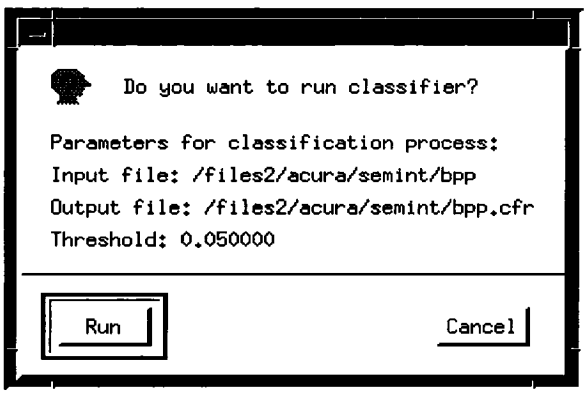

Figure 6: Classifier Window

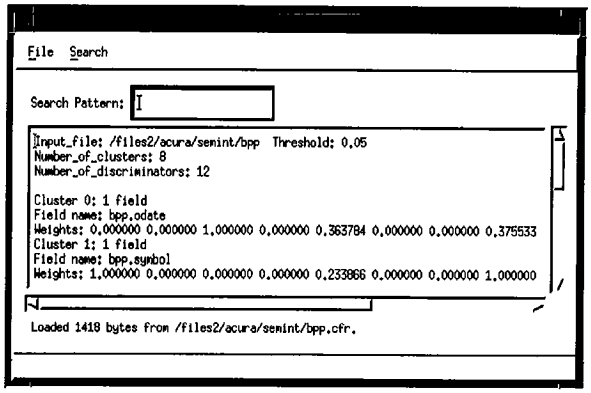

Figure 7: Classifier Output

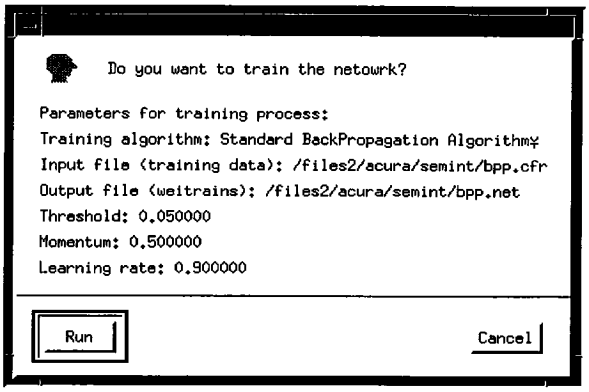

Figure 8: Training Window

Training Networks

The output of the classifier is then used as training data for a back-propagation network, a *supervised* learning algorithm. The "supervision" is that target results are provided; however as these target results are the output of the classifier, no user supervision is needed. We use this as a training algorithm to train a network to recognize input patterns and give degrees of similarity. We have implemented two versions of back-propagation algorithms, standard and quick-propagation. For details of the training algorithms, please see [Li and Clifton, 1994]. The window dump for training is shown in Figure 8.

Using Networks

In order to determine similarity between two databases, users take the network trained for one database (e.g. database A), and use information extracted from the other database (e.g. database B) as input to this network. The network then gives the similarity between each pair of attributes in the two

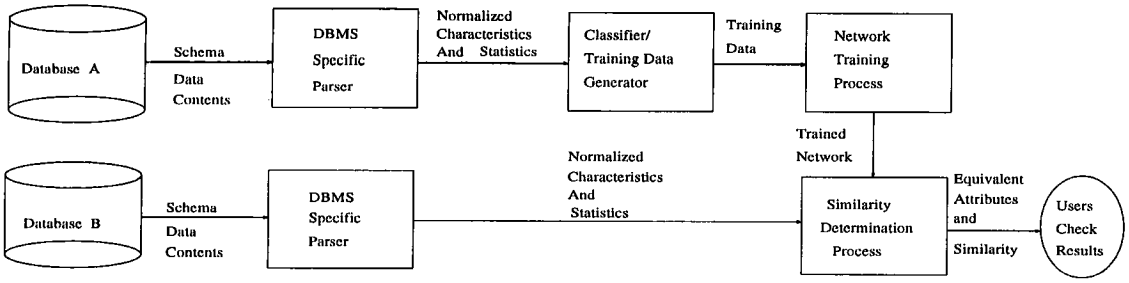

Figure 9: Overview of Semantic Integration Procedure in Semint

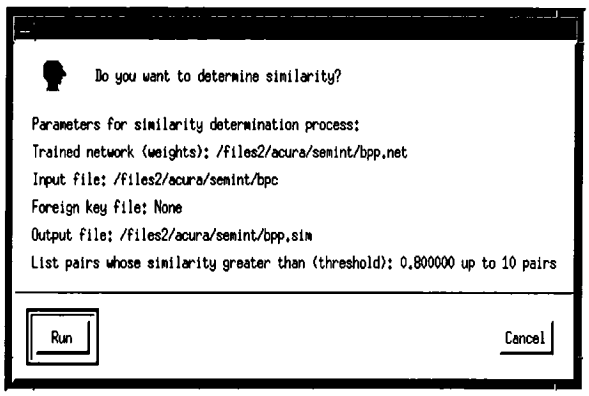

Figure 10: Similarity Determination Window

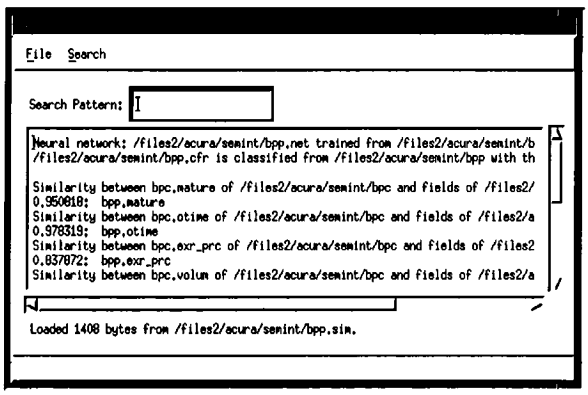

Figure 11: Similarity Output

databases. System users check and confirm the output results of the trained network. This process is shown in Figure 9. A window dump of similarity determination and its output are shown in Figures 10 and 11. The output results include lists of similar attributes and the similarity between them. Note the only human input is to specify DBMS types and IP addresses for remote machines and to examine and confirm the output results (similar pairs of attributes and the similarity between them). Other processes can be fully automated. Given DBMS types, user id, password, database owner id, and machine IP addresses. Semint will automatically extract schema information from DBMS, analyze data contents to generate statistics, transform database information into a single format, build and train neural networks, and then use trained neural networks to identify similar attributes and determine their similarity. The source code of current system is available through anonymous FTP from eecs.nwu.edu in /pub/semint.

Domain Knowledge and Detecting Heterogeneity

Some data items, such as first names, last name, city, date, and time, frequently exist in databases. Using specialized functions with domain knowledge can identify these data items independently. For example, when we examine a data item "departure", its field name only tells us this data item is related to "departure". We then find some city names in this field, say, "Chicago" or "ORD", we may conclude "departure" records departure city, not departure date or time. On the other hand, if we find the contents in this field are dates, we may come up a different conclusion - "departure" records departure date, not departure city. Also, if we find some first names such as "John" existing in two fields, we may conclude these two fields are "first name" even if their field names are not assigned with meaningful names and their schema is designed differently, say, Char(10) and Varchar(20). Integration knowledge base is not necessarily something that generalizes, so learning in a specific domain knowledge base (e.g. a domain knowledge base for finance related databases) can be quick, easy, and less expensive to build (as opposed to trying to come up with a monolithic procedure that works for all cases).

Problems of heterogeneity are pointed out by

[Wiederhold, 1993] such as representation, scale, and level of abstraction. [Sciore *et al.*, 1994] shows how semantic values can be either be stored explicitly or be defined by environments using conversion functions (e.g. *4(LengthUnit='feet')* and *48(LengthUnit='inches')*). Heterogeneity can be "resolved" because there exists *conversion* between them. Thus, heterogeneous databases can be accessed using an extension of SQL, called Context-SQL. However, the main problem of overcoming the challenges of large-scale interoperable database systems, finding similar data items, remains since conversion functions need to be defined manually and standardized.

We are combining the similarity determined in all levels to identify types of heterogeneity. If we compare two fields which are semantically equivalent but use different units. We may come out a result that their names are similar in the dictionary level, their data types are the same but length is different in the schema level, and their average values are different but their CV's are similar in data content level. We may conclude there exists scale heterogeneity (e.g. one attribute uses "pound" while another uses "kg").

Conclusion

A number of federated database system prototypes have been or being developed. These systems focus on query processing and transaction management assuming heterogeneity has been identified and resolved so that global queries can be decomposed into local queries. In these systems, identifying heterogeneity needs to be done manually and equivalence relationship maps are specified by "user-defined functions". Semint can be used to identify heterogeneity and build equivalence relationship maps (output results of similarity determination process from Semint) for these systems as a pre-processor. [Wiederhold, 1993] states important issues in intelligent integration of information include metadata representation and how to match users' specifications with huge amount of available resources. In this paper we give the process of knowledge extracting and matching in Semint. We argue trying to come up with a general framework for dealing with heterogeneity will probably fail, but learning techniques enable us to come up with specific frameworks for localized problem domains without a great deal of human involvement. Efforts in building domain knowledge bases to identify frequently appearing data items such as names, address, SSN, company names, and airports, will be compatible with and support current techniques in identifying semantically related data items.

References

Drew, P.; King, R.; McLeod, D.; Rusinkiewicz, M.; and Silberschatz, A. 1993. Report of the workshop on semantic heterogeneity and interoperation in multidatabase systems. *SIGMOD Record* 47–56.

Goh, Cheng Hian; Madnick, Stuart E.; and Siegel, Michael D. 1994. Context interchange: Overcoming the challenges of large-scale interoperable database system. In *Proceedings in the 3rd International Conference on Information and Knowledge Management.* ACM. 337–346.

Li, Wen-Syan and Clifton, Chris 1993. Using field specification to determine attribute equivalence in heterogeneous databases. In *Third International Workshop on Research Issues on Data Engineering: INTEROPERABILITY IN MULTIDATABASE SYSTEMS*, Vienna, Austria. 174–177.

Li, Wen-Syan and Clifton, Chris 1994. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of 20th International Conference on Very Large Data bases*, Santiago, Chile. 1–12.

Li, Wen-Syan and Clifton, Chris 1995. Semint: A system prototype for semantic integration in heterogeneous databases. to appear in *Proceedings of 1995 ACM SIGMOD Conference*, San Jose, CA. Prototype demonstration.

Premerlani, William J. and Blaha, Michael R. 1994. An approach for reverse engineering of relational databases. *Communications* 37(5):42–49.

Sciore, E.; Siegel, M. D.; and Rosenthal, A. 1994. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Databases* 19(2):254–290.

Ventrone, Vincent and Heiler, Sandra 1994. Some advice for dealing with semantic heterogeneity in federated database systems. In *Proceedings of the Database Colloquium*, San Diego. AFCEA.

Wiederhold, Gio 1993. Intelligent integration of information. *SIGMOD Record* 434–437.