SOURCE DISCOVERY AND SCHEMA MAPPING

FOR DATA INTEGRATION

by

Li Xu

A dissertation submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Brigham Young University

October 6, 2003

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a dissertation submitted by

Li Xu

This dissertation has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	David W. Embley, Chair
Date	Stephen W. Liddle
Date	Deryle W. Lonsdale
Date	Dan R. Olsen
Date	Scott N. Woodfield

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the dissertation of Li Xu in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

David W. Embley
Chair, Graduate Committee

Accepted for the Department

David W. Embley
Graduate Coordinator

Accepted for the College

G. Rex Bryce
Associate Dean
College of Physical and Mathematical Sciences

ABSTRACT

SOURCE DISCOVERY AND SCHEMA MAPPING

FOR DATA INTEGRATION

Li Xu

Computer Science

Doctor of Philosophy

As data explodes on the Web, there is a need to integrate data from a large number of heterogeneous information sources. Currently, there are two main basic approaches to data integration: Global-as-View (GAV) and Local-as-View (LAV). However, both approaches have their limitations for large-scale applications. To resolve the problems, we offer a Target-based Integration Query System (TIQS) as an alternative point of view that is neither GAV nor LAV. The approach uses a predefined conceptual target schema, which is specified ontologically and independently of any of the sources, as a central, organizing concept. In this dissertation, we focus on the resolutions to three problems in TIQS: (1) automatically recognizing information sources for the target, (2) automating source-to-target mappings between source and target schemas, and (3) query reformulation based on source-to-target mappings. Experiments we have conducted show that we have been able to achieve good performance for the recognition of applicable documents as well as the generation of source-to-target mappings. Moreover, we have proven that query reformulation in TIQS reduces to rule unfolding and the reformulated user queries extract all the query answers available from sources with respect to the definition of TIQS for the proposed queries.

ACKNOWLEDGMENTS

I thank God who gave me the intelligence to finish this dissertation.

I thank my parents, who made sacrifices to ensure that I had a good education. I thank my sister, who encouraged and supported me to continue my study in America. I deeply thank my husband, Yihong for his patience, his love, and his suggestions in my study. My lovely daughter, Vania, brings me much joy.

I am very grateful to my mentor, Dr. David W. Embley. It was with Dr. Embley that I first learned how to do research, how to write papers, and how to present technical materials. I, especially, thank Dr. Embley for always being warm and caring and for putting in encouraging words when I felt down.

I appreciate the support of Dr. Stephen W. Liddle and Dr. Deryle Lonsdale, who have been very helpful in shaping many of the ideas that comprise this research work, and enhancing my English.

I am grateful to Dr. Dan R. Olsen, and Dr. Scott Woodfield for their feedback on this research and for being on my dissertation committee.

I thank Cui Tao, Xueqi Chen, and David Jackman. They have spent countless hours discussing the research with me. Especially Cui and Xueqi helped me collect the experimental data for part of my research. Their help was essential to the success of this research.

The National Science Foundation, grant number IIS-0083127, provided partial financial support for this research work.

Finally, I would like to thank all the people in the Data-Extraction Research Group. In particular, I thank Troy Walker, who spent a lot of time to read this dissertation and provided invaluable feedback to me. I would not have been able to accomplish this work without their contributions.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In open and evolving environments such as the World Wide Web, the amount of information available is proliferating at a tremendous rate. Web data comes in various forms. At one extreme, online traditional database systems manage large amounts of highly structured data, much of it behind form interfaces. At the other extreme, many Web pages are totally unstructured (containing raw text written in natural language), or at least only informally semistructured (containing some elements of typed or structured data presented in an unstructured or semistructured way).

With all of this data, one of the huge challenges we face is the integration of data that is extracted from structured, semistructured, and unstructured sources pertaining to a domain of interest. Our approach to this problem begins with a domain-of-interest specification, which we call a *target schema*. We define a target schema as a conceptual-model instance like the ontology in [21]. Such an ontology consists of a description of object sets and relationship sets among the object sets that are of interest to the end user (the creator of the ontology). An ontology may also include constraints and reasoning or inference rules that further describe the domain of interest. Given a particular target schema, we can integrate multiple information sources by (1) discovering applicable information sources; (2) individually matching and reconciling the concepts in applicable information sources with corresponding concepts in the target schema; and (3) merging the related data. The target schema is a central, organizing concept for our approach.

The objective of this dissertation is to resolve the problems of discovering applicable information sources and then matching and reconciling information sources with a target schema.[1] Significantly, a target schema is independent of any particular information source.

Given a target schema, we address the following two subproblems:

1. *Source discovery.* We first determine whether an information source or any portion of the source is applicable to the target schema.

2. *Source matching and reconciliation.* Given a set of applicable sources, it is likely that individual sources will use different object sets, different terminology for the same object sets, different relationship sets, or different values with the same object sets. Therefore we must match elements of the various applicable sources with the target schema and reconcile these differences.

Solving these two problems will allow us to identify data of interest and bring it into a target schema, a common formalism, so that the data can be fully integrated (assuming that the downstream problem of merging related data is resolved). Before giving the details about how we solve those two problems, we first give an overview of related work in source discovery and data integration.

1.1 Source Discovery

Discovering desired information is related to information retrieval (IR) [30, 63]. IR is concerned with indexing a collection of documents based on document topics, and answering queries by returning a ranked list of relevant documents [2]. In particular, the focus is on measuring the similarity of two documents or the relevance of documents to a given request. In contrast, the central focus for discovering desired information in our approach is to identify documents that contain objects and relationships compatible with those specified by a desired target schema. Thus, the discovered information not only relates to the topic of a particular application, which is similar to the goal of the IR problem, but also relates to the structure of the target schema. Moreover, the object sets and relationship sets comprise a finer level of information than the application topics used in the IR approach.

[1] These problems are difficult enough that the downstream problem of merging related data is deemed to be beyond the scope of this dissertation.

Given this finer level of information structure, the data contained in an information source can be extracted according to the specification in a target schema.

The classical approach of IR is to rely on keyword matching. That is, keywords are extracted from each document either manually or automatically, and documents are compared based on the extracted keywords. Interrelationships among keywords are usually ignored. For example, while a classical IR approach might classify one document containing a sentence like "Wilhemine Friderike Carolina ERDMAN, born on 30 OCT 1864, Buffalo, Erie, NY," it will not consider the relationship between "Wilhemine Friderike Carolina ERDMAN" and "Buffalo, Erie, NY." Thus, classical IR systems cannot precisely find applicable information sources that match a target schema containing relationships like "person was born at place". This is because keywords do not indicate which object-set patterns and relationship-set patterns are compatible with the specifications declared in a target schema. Classical IR systems are good for preliminary exploration, but they retrieve too many irrelevant information sources for our purposes.

We can provide a better source discovery solution by using all the information in the target schema specification. Given the values recognized by the target schema specification and metadata extracted from an information source, we can construct a set of applicability heuristics. Then, based on machine-learned rules over these heuristic measurements, we determine whether an information source is applicable to a given target schema. Our approach can markedly improve precision over the keyword-based IR approaches.

Our approach for source discovery largely utilizes machine learning techniques to exploit a broad set of heuristics in order to produce rules to detect applicable information sources. Modern IR systems also include learning-based text classification [8, 38] as an IR task. Text classification often depends on keyword-based techniques and statistical learning methods to classify document topics. While applying text-classification methods in IR systems, one or more category labels are assigned to a document. This method presumes a predefined, static set of user interests (categories) but suffers because of a lack of semantic analysis. In contrast to our approach, text-classification methodologies cannot process complex information requests specified as target schemas.

Information extraction techniques based on natural language processing (NLP) have been shown to be effective for high-precision text classification [61], which is able to deal with more complex information requests than traditional text classification. However, NLP techniques are computationally expensive, especially for user requests as complex as the specifications declared in a target schema. Thus, it is difficult for NLP approaches to scale up to the large quantity of documents in the context of our source discovery and data integration problem. In contrast, our approach depends on ontology-based data extraction techniques [21] to recognize values from an information source, so our data-extraction technique is not as expensive as NLP alternatives.

Ontology-based data extraction techniques maintain knowledge for an application domain in the specification of a desired target schema. Similarly, within the IR community, knowledge-based IR systems [33, 36, 60] have been developed that address some problems raised in keyword-based IR systems using rule bases or other knowledge resources. Knowledge-based IR systems have attempted to capture searchers' and information specialists' domain knowledge and classification scheme knowledge. However, they usually do not have learning ability and perform only what they are programmed to do. Moreover, most knowledge-based IR systems require an extensive manual knowledge-engineering effort that takes significant time and human resources to acquire knowledge from domain experts. In contrast, our approach has the ability to learn the knowledge needed to do the classification and thus is much easier to port to new application domains.

1.2 Data Integration

In our approach, we attack the problem of source matching and reconciliation in the context of data-integration systems. In the following, we give an overview of data-integration systems as well as approaches to the problem of source matching and reconciliation.

From a practical point of view, an important distinction in building a data-integration system is whether to take a warehousing or a virtual approach [40]. In the warehousing approach [46, 76], data from multiple information sources is loaded into a warehouse, and user queries are applied to the data warehouse. The warehouse approach requires updates to the warehouse when the source data changes and the updates are typically done in batches, not on demand. Such an approach guarantees adequate query performance because queries are read-only and operate on a single repository. In the virtual approach, the data remains in the information sources, and user queries are decomposed at run time into queries on the information sources. The virtual approach is appropriate when the number of information sources is large and individual information sources change frequently, but it requires more sophisticated query optimization and execution methods to guarantee adequate performance.

From a theoretical point of view, the traditional approach to data integration is to create a global schema, which is a set of object sets and relationship sets that describe individual entities and relationships between the entities in the domain of expected user queries. To evaluate queries, the data-integration system translates queries on the global schema into queries on the underlying information sources. There are two basic approaches [71]: Global-as-View (GAV) [4, 17, 35, 41, 66, 69] and Local-as-View (LAV) [1, 29, 32, 42, 50, 64].

In the GAV approach, global concepts, which consist of object sets and relationship sets found in the sources, are synthesized in the global schema. In the LAV approach, however, global concepts are declared independent of information sources. The main advantage of the GAV approach is that query reformulation is simple, because it reduces to rule unfolding, which means that queries to the sources are immediately available based on the view specification in a global schema. However, whenever a new information source is added or a current underlying information source is changed, the change needs to be propagated to the global schema. In contrast, in the LAV approach, it is simple to add or delete information sources since the source descriptions do not need to take into account the interactions with other source descriptions. But it is difficult for an LAV data-integration system to process complicated queries because the LAV approach has to apply query-rewriting algorithms to decompose a user query into subqueries in terms of the information sources. The query-rewriting algorithms generate the rewriting in time that is exponential in the size of the query and view definitions in the source descriptions.

Our target-based approach complements both the GAV and the LAV approaches. Like GAV, our target schema provides a global schema in terms of global concepts and the interrelations between the concepts, but like LAV, the global schema is declared independent of the information sources. The independent target schema limits the scope of the integration to a predefined set of concepts. This makes the integration work more manageable than with the traditional GAV approach and keeps the scalability as in an LAV approach. Further, each information source has its own independent source declaration, which is a populated conceptual schema. We map each source schema to the target schema in such a way that query formulation results in rule unfolding. This allows the integration to have better query response time than with the traditional LAV approach. The cost of our target-schema approach is that we need to manually construct a target schema and produce a mapping between the target schema and each of the information sources. Our experience in teaching others to construct target schemas suggests that a target schema for an application such as automobile want-ads can be created in a few dozen person-hours.

Work on generating a mapping between a target schema and an information source can be classified into rule- and learner-based matching approaches. Rule-based matching approaches usually utilize only schema information and normally only in a hard-coded way [14, 47, 54, 56]. Whenever the system changes to a new application domain, the hard-coded rules must be changed dependent on domain constraints. Learning-based matching approaches apply machine learning techniques to exploit a broad set of properties utilizing either schema or data-instance information [18, 19, 43, 52]. When the system changes to a new application domain, it requires supervisor-provided training data in order to train a learner to match schemas for the new application. We construct a composite approach that exploits a set of similarity measurements between the object and relationship sets of a target schema and the object and relationship sets of a source schema. The composite approach generates a source-to-target mapping, which maps a source schema to a target schema, using machine-learned rules, domain ontologies, and rules based on structural properties of schemas. A source-to-target mapping determines whether an object or relationship set (given or derived) in the source schema can match an object or relationship set in the target schema. Compared with rule-based matching approaches, our approach is easier to port to a new application domain because the learning methods applied in our approach automatically generate the matching rules. Compared with learning-based matching approaches, the machine-learned rules applied in our approach are universal such that we do not need to train the learners to match schemas when porting to new application domains. Moreover, most of the existing approaches focus on discovering only direct matches between two schemas. Our approach, however, discovers both direct and many indirect matches between source and target schemas.

1.3 Dissertation Statement

We can resolve several problems with existing approaches to information discovery and integration by starting with a target schema specified as an ontology that is conceptual and independent of any information source. We can develop semi-automatic procedures that use a given target schema T to (1) identify whether an information source S applies to T, and (2) generate a mapping between S and T. Generated mappings allow data to be extracted from applicable sources so that queries processed against the target can be applied to a wide range of applicable sources. Our approach constitutes an improved solution because our method (1) can process more complex user requests (specified in a single target schema), (2) can discover applicable information sources more precisely, (3) is more scalable and portable than IR systems, (4) can discover both direct and many indirect matches between S and T, and (5) retains both the query performance of the GAV approach and the scalability of the LAV approach to unify a large number of heterogeneous information sources.

1.4 Dissertation Overview

The body of this dissertation consists of two manuscripts for journal publication (Chapters 2 and 3). Each, therefore, has its own abstract, introduction, and conclusions. There is also a small amount of intentional overlap between the two, so that each is self-contained.

Chapter 2 describes an approach to recognize applicable HTML documents for a particular application. Automatically recognizing which HTML documents are of interest for some specified application is non-trivial. As a step toward solving this problem, we propose a technique for recognizing which HTML documents apply to an ontologically specified application. The approach largely exploits the specification of a target schema and machine learning techniques. Our experimental results show that we have been able to achieve above 95% precision, recall, and F-measures.

Chapter 3 describes an approach to automate schema mapping for large-scale data-integration applications. Automating schema mapping is challenging. Previous approaches (e.g. [19, 47]) to automating schema mapping focus on computing direct matches between two schemas. Schemas, however, rarely match directly. Thus, to complete the task of schema mapping, we must also compute indirect matches. In this dissertation, we present a framework for generating a source-to-target mapping containing direct and many indirect matches between a source schema and a target schema. Recognizing expected data values associated with schema elements and applying schema-structure heuristics are the key ideas to computing indirect matches. Experiments we have conducted over several real-world application domains show encouraging results, yielding over 90% precision and recall measures for both direct and indirect matches. Based on source-to-target mappings, we offer a unified approach for data integration. Compared with other data-integration approaches, our data-integration approach combines their advantages, mitigates their disadvantages, and provides a viable alternative for flexible and scalable data integration.

In Chapter 4, we draw conclusions. We also propose future work, which we can do based on this dissertation.

Chapter 2

Recognizing Ontology-Applicable HTML Documents

Abstract: Automatically recognizing which HTML documents are "of interest" for some specified application is non-trivial. As a step toward solving this problem, we propose a technique for recognizing which HTML documents apply to an ontologically specified application. In this chapter, the HTML documents we consider include semistructured HTML documents, HTML tables, and HTML forms. Given the keywords and values and kinds of values recognized by an ontological specification in an HTML document, we apply several heuristics: (1) a set of density heuristics that measure the percent of the document that appears to apply to the application ontology, (2) an expected-value heuristic that compares the number and kind of values found in the document to the number and kind expected by the application ontology, and (3) a grouping heuristic that considers whether the values of the document appear to be grouped as application-ontology records. Then, based on machine-learned rules over these heuristic measurements, we determine whether an HTML document is applicable for a particular application. Our experimental results show that we have been able to achieve about 95% for both recall and precision.

Keywords: Document/ontology matching; binary classification; database-centric information retrieval; machine-learned classification rules; ontological specification; data extraction; information filtering, domain-specific search.

2.1 Introduction

The World Wide Web contains abundant repositories of information in HTML documents—indeed, it contains so much that locating information "of interest" for an application becomes a huge challenge. Even sorting through a tiny subset of HTML documents is overwhelming. How can we automatically select just those documents that have the needed information for an application?

When we construct automated processes to recognize which documents apply to a user's information needs, we must be careful not to discard relevant documents and not to accept irrelevant documents. A process that discards too many relevant documents has poor *recall*—the ratio of the number of relevant documents accepted to the total number of relevant documents. A process that accepts too many irrelevant documents has poor *precision*—the ratio of the number of relevant documents accepted to the total number of documents accepted. The harmonic mean of the precision and recall, which is called an *F-measure* is a standard way to combine precision and recall. We wish to have an automated recognition process that has both high recall and high precision.

In this chapter, we propose an approach for recognizing whether a Web document is relevant for an application of interest. We base our approach on application ontologies [21], which are conceptual-model snippets [67, 72] of standard ontologies [9, 10], and we apply techniques from information retrieval [2, 63] and machine learning [58].

Our work reported here is also partly motivated by our success in using application ontologies to extract and structure information from HTML documents so that it can be queried using a standard query language [21]. For several applications we have tried—automobile want-ads, obituaries, jobs, real estate, stocks, etc.—we have achieved fact-extraction recall rates mostly around 90% and fact-extraction precision rates mostly better than 90%, and we have achieved robustness over a wide range of pages that change in format, content, and style [21, 26, 25]. In these experiments, however, we have assumed (and have made sure by human inspection) that the Web pages were appropriate for the application we were using. Thus, in the context of our larger project, the purpose of this work is to automate applicability checking. If we can locate documents applicable to an ontology, we can apply techniques we have already developed, to extract, structure, query, and archive in databases, information found in data-rich, application-specific HTML documents. Hence, the contributions of this work have the potential to be more far-reaching than just the salient contribution of increasing F-measure, recall, and precision in recognizing application-specific HTML documents.

Our approach to document recognition is related to text classification [65]—each application ontology can be a class—but our work fundamentally differs from other text-classification work. Text classification systems usually attempt to place documents such as newspaper articles in predefined classes according to the subject matter of the article, whereas our approach seeks to do "high-precision" text classification (with similarities to [61]) in which we not only determine whether a listing of ads such as the classified ads in a newspaper contains ads of interest for a predefined application ontology, but we also determine whether particular elements of interest are also present in each ad. We further assume that a subsequent process can extract the information and create a database record for each ad.

Despite these basic differences, we nevertheless compare our work with the work in text classification (e.g. [16, 65]) in order to highlight some advantages and disadvantages of the approaches that have been taken. Most text classification systems are based on machine learning. In typical machine learning approaches, each document is transformed into a feature vector. Usually, each element of a feature vector represents a word from a corpus. The feature values may be binary, indicating presence or absence of the word in the document, or they may be integers or real numbers that indicate some measure of frequency of the word's appearance in the text. This text representation is referred to as a "bag of words," which is used in most text classifiers. A major difficulty for this bag-of-words approach is the high dimensionality of the feature space. Thus it is highly desirable to reduce dimensionality of the space without sacrificing classification accuracy. Since our approach uses a predefined application ontology whose object sets constitute the features of interest, we immediately identify a space with comparatively small dimensionality and thus avoid this high-dimensionality problem. Further, our predefined application ontology also overcomes many of the limitations imposed by word-based techniques. Our approach is sensitive to context and domain knowledge and can thus more effectively retrieve the relevant information from a document and use it to classify a document with respect to an application. Moreover, there is no need to find object relevancy with respect to a corpus because the application ontology already defines the relationships among the conceptual objects. In compensation for these disadvantages, typical machine learning approaches may take less user effort to produce—the effort being mainly the work to provide and label a set of training documents rather than this effort plus the effort to create an application ontology as required by our approach. For example, the basic idea of McCallum's Naïve Bayes classifier [3], which is implemented in Rainbow [48], is to use the joint probabilities of words and categories, which are computed based on the labeled training documents, to estimate the probability of a given document belonging to a category. However, the naïve part of the approach is the assumption of word independence, which makes the classifier inappropriate for producing "high-precision" classifications like ours.

Although our work differs fundamentally from most text classifiers, the work reported in [61] takes an approach similar to ours in that it also attempts to do "high-precision" classification for information extraction. Like most text classifiers, [61] uses machine learning, but to obtain the desired high precision, considerably more effort must be expended to establish the basis for machine learning. Not only must documents be marked as relevant and non-relevant, each individual relevant element plus the context for each individual relevant element must also be marked. In addition, an application-domain-specific dictionary must be created. The basic tradeoff in human effort between the approach in [61] and our approach is the effort to tag the documents as well as the elements in the documents and to create the domain-specific dictionary versus the effort to tag the documents (but not the elements) and to create the application ontology.

Some recent work has been reported that uses machine learning with less human effort for doing "high-precision" classification for domain-specific search engines [49]. By using unsupervised learning, human effort can be greatly reduced. The challenge, however, is to reach high accuracy, and it may not be possible to achieve the accuracy that can be obtained with an ontology-based approach. Ultimately, some combination of the approaches may be best. In the meantime, we pursue our goal of high-precision binary classification based on ontological specifications.

We outline the rest of our paper as follows. Section 2.2 presents some preliminaries and provides an example to which we refer throughout the paper to illustrate our ideas. Section 2.3 describes the high-level architecture of the framework we build to recognize applicable HTML documents for application ontologies. Given an application ontology and a set of HTML documents, Section 2.4 explains how we automatically obtain statistics for a set of heuristics for each document: (1) a set of density heuristics, (2) an expected-values heuristic, and (3) a grouping heuristic. Section 2.5 provides an empirical evaluation of our approach including our experimental results which—for the two applications we tried (car advertisements and obituaries)—have measures (precision, recall, and the F-measure) of 95%. Section 2.6, gives concluding remarks and our future-work plan.

2.2 Preliminaries

Before giving details of our approach, we first need to discuss our assumption about HTML documents, application ontologies, and the classification task. Throughout this chapter we use car ads as a running example. Thus, the illustration HTML documents and application ontologies are all about cars for sale.

2.2.1 HTML Documents

In this dissertation, the HTML documents on the Web we consider include semistructured HTML documents such as the lists of ads in Figure 2.1, HTML tables such as the one in Figure 2.2, and HTML forms such as the one in Figure 2.3(a), which when extracted manually, yields a table or a semistructured list such as the one in Figure 2.3(b).

We assume that the HTML documents are data-rich and narrow in ontological breadth [21]. A document is *data rich* if it has a number of identifiable constants such as dates, names, account numbers, ID numbers, part numbers, times, currency values, and so forth. A document is *narrow in ontological breath* if we can describe its application domain with a relatively small ontological model. The documents in Figures 2.1–2.3 are all data rich and narrow in ontological breadth.

When evaluating the applicability of HTML documents to a particular application, we want to exploit the contents rather than HTML tags and layout features. Thus,

Last Updated
Monday, January 24, 2000 12:19pm Cars for Sale

DEPENDABLE CAR
1989 Subaru SW. Auto, AC, $1900 OBO. Call (336)835-8579.

FACTORY WARRANTY
1998 Elantra. Black 4 door w/ tinted windows. Auto, pb, ps, cruise, am/fm cassette stereo, a/c. Excellent condition pay off OBO. Call (336)526-5444 anytime & leave message.

1994 HONDA ACCORD EX
Auto, power everything, jade green w/gold package. Under 100K miles. Call (336)526-1081 after 7pm.

1999 GRAND AM
27,000 miles, silver, auto, still under warranty. $14,000 OBO. Call (336)366-4996 anytime.

`53 Chevy Bel Aire. All original, looks like new. Serious inquiries only. $8500. Call (336)468-8924 after 4 pm.

TWO GREAT CARS
1973 MGB convertible. British racing green. Mags, new tires, 4-speed, 1 owner, excellent running condition. $4500.
1977 Olds Cutlass Supreme. New white paint job w/ 1/2 red Landau top, original mags & new tires. Auto., 1 owner, low mileage, loaded. Call (336)984-2843.

----

95 FORD CONTOUR
5-speed, great condition, one owner, $5300. Call (336)526-8853 & leave message if no answer.

SEIZED CARS FROM $500
Sports, luxury & economy cars, trucks, 4x4's utility and more. For current listings, call 1-800-311-5048 ext. 10012.

1996 VW JETTA GL
26,000 miles. 4 door, 5-speed, AC, sunroof, 1 owner. $11,000. Call (336)874-7317 anytime.

`85 Buick Park Avenue. $500. Head may be cracked. Will run. Body good condition. Call (336)526-2768.

`95 Ford Thunderbird. Loaded, V-8, 45K, $6995. Call S&J Motors at (336)874-3403.

`96 Mercury Tracer. 4 door, 5 speed, 34K, $4995. Call S&J Motors at (336)874-3403.

`88 Firebird. V8, 5.0, fuel injected, T-tops, 109,000 miles, red, runs great. $1880. Call (336)526-1164 anytime.

1990 CONVERSION VAN
350 motor, auto, new tires, TV, VCR, captain chairs, front & rear AC. $4,995. Call (336)320-2658 anytime.

COMMERCIAL WORK VAN
`95 Chevy Astro, V6, w/ac & fully equipped utility shelves. $9400. Call (336)526-2675 & leave message.

(a) Car advertisements retrieved from http://www.elkintribune.com/

Last update: Wednesday, December 22, 1999

Select a category

Apartment For Rent For Sale or Rent Lost or Found
For Rent Help Wanted
For Sale House For Rent

Apartment For Rent
ONE EFFICIENCY, 2 & 3 bdrm, all utilities paid. Call 281-2051 -

----

For Rent
HOUSING SOLUTIONS - Free TV cable furn. $60/wk - $210/mo. 281-4060. -

----

For Sale
1998 JD 455 mower, 60' deck. Call for price. Also, homemade Go-Cart. Call after 5:30 pm 218-281-1128. -

----

For Sale or Rent
10,000 SQ. FT. office building. Handicap accessible. Call 281-3631. -

----

Help Wanted
NOW HIRING full time and part time customer service representatives. Advancement possible and weekly pay. Must be able to work weekends and holidays. Apply at Superamerica, 411 N Main St., Crookston, MN EOE -

----

PART-TIME AND weekend help working with developmentally disabled adults. Call Melissa or Karen at 281-3872. -

----

REM-NORTHWEST Services, Inc. has a full time Program Coordinator/Coordinator position open in Crookston working with four developmentally disabled adults. Duties include hiring, staff supervision, scheduling, oversight of most areas of the home's operation. Applicant must be 18 years of age or older. Must have a high school diploma or equivalent. One year experience serving people with developmental disabilities preferred. Must have a valid driver's license and driving record that meets REM's insurability requirements. Insurance and benefits available. If interested call for application at 218-281-5113. E.O.E. -

----

REM-NORTHWEST Services, Inc. has full and part time Coordinator positions available in Crookston, MN, working with citizens who are developmentally disabled. Excellent benefits are offered including health, dental, life, 401K and profit sharing for full and part time employees working 20 hours a week or more. Exceptional training is provided. Applicants must be 18, have a valid driver's license and high school diploma or GED. Apply by calling for application at 218-281-5113 or 1-800-532-7655. E.O.E. -

----

House For Rent
3 BDRM HOUSE $450/mo. 281-1970. 22 STEEL BUILDINGS, NEW, must sell. 40x60x14 was $17,500 now $10,971; 50x100x16 was $27,850 now $19,990; 80x135x16 was $79,850 now $42,990; 100x175x20 was $129,650 now $78,850. 1-800-406-5126. -

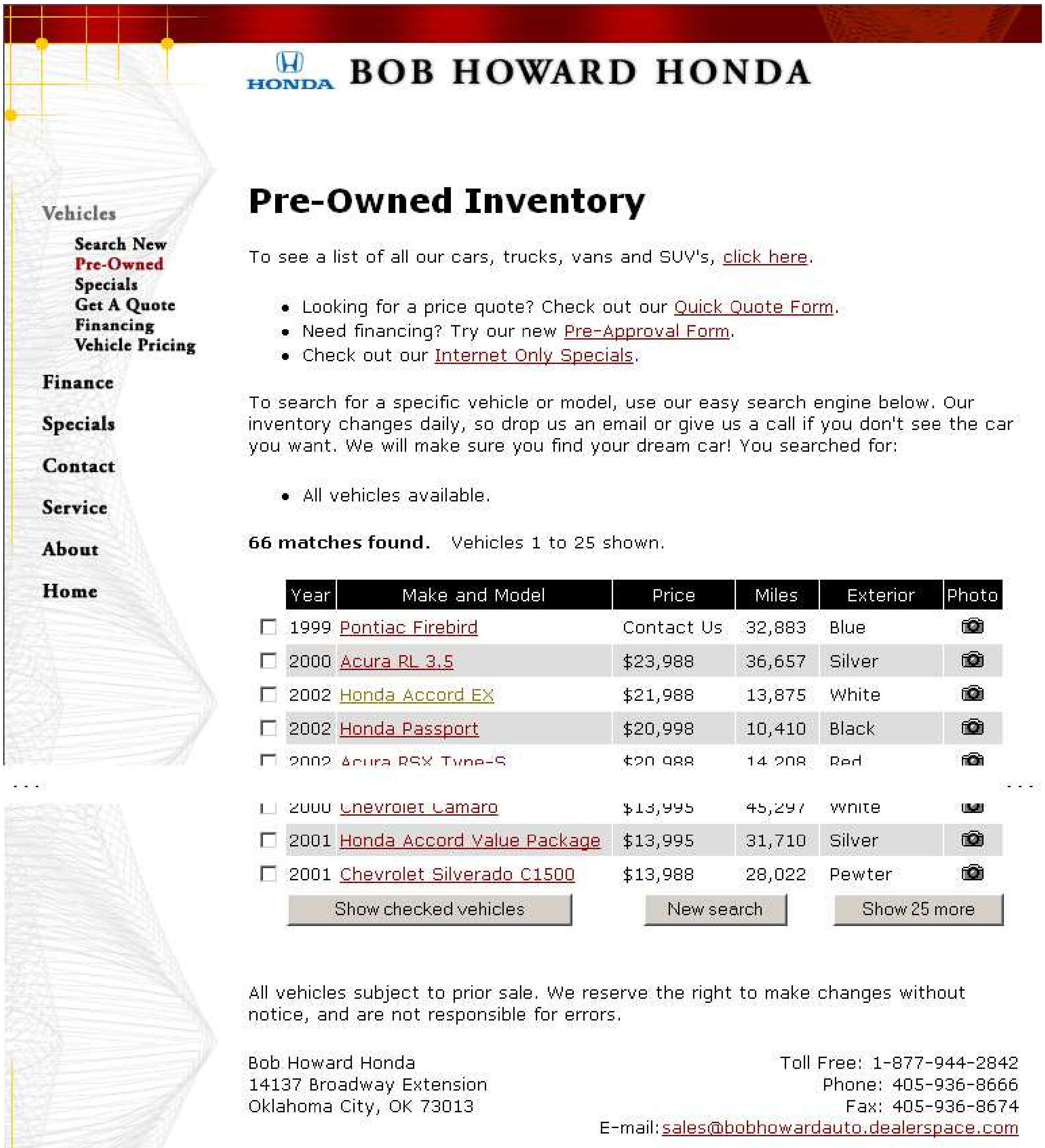
----

Lost or Found
FOUND: Golden retriever about 4 months old. Found 7 miles south of Crookston. Call after 5:30 pm 281-1128. -

----

(b) Items for sale advertisements retrieved from http://www.crookstontimes.com

Figure 2.1: A car-ads Web document and a non-car-ads Web document



BOB HOWARD HONDA

Vehicles
Search New
Pre-Owned
Specials
Get A Quote
Financing
Vehicle Pricing
Finance
Specials
Contact
Service
About
Home

Pre-Owned Inventory

To see a list of all our cars, trucks, vans and SUV's, click here.

- Looking for a price quote? Check out our Quick Quote Form.
- Need financing? Try our new Pre-Approval Form.
- Check out our Internet Only Specials.

To search for a specific vehicle or model, use our easy search engine below. Our inventory changes daily, so drop us an email or give us a call if you don't see the car you want. We will make sure you find your dream car! You searched for:

- All vehicles available.

66 matches found. Vehicles 1 to 25 shown.

	Year	Make and Model	Price	Miles	Exterior	Photo
☐	1999	Pontiac Firebird	Contact Us	32,883	Blue	
☐	2000	Acura RL 3.5	$23,988	36,657	Silver	
☐	2002	Honda Accord EX	$21,988	13,875	White	
☐	2002	Honda Passport	$20,998	10,410	Black	
☐	2002	Acura RSX Type-S	$20,988	14,208	Red	
...						
☐	2000	Chevrolet Camaro	$13,995	45,297	White	
☐	2001	Honda Accord Value Package	$13,995	31,710	Silver	
☐	2001	Chevrolet Silverado C1500	$13,988	28,022	Pewter	

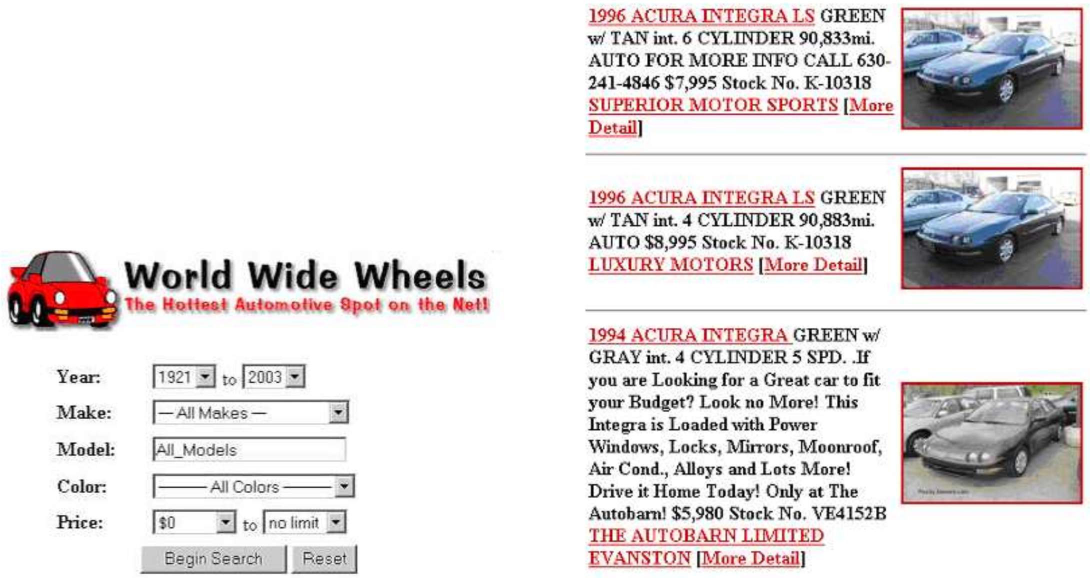
Show checked vehicles | New search | Show 25 more

All vehicles subject to prior sale. We reserve the right to make changes without notice, and are not responsible for errors.

Bob Howard Honda
14137 Broadway Extension
Oklahoma City, OK 73013

Toll Free: 1-877-944-2842
Phone: 405-936-8666
Fax: 405-936-8674
E-mail: sales@bobhowardauto.dealerspace.com

Figure 2.2: HTML page with table at [7]



(a) HTML form at [74]

(b) Retrieved car ads after filling in form in Figure 2.3(a)

Figure 2.3: HTML form at [74]

even though the designers of the HTML documents express the contents in various ways, we recognize the documents applicable to the application mainly based on three patterns: *multiple-record* documents, *single-record* documents, and *application forms*, which we distinguish by the contents with respect to the target application. The documents in Figures 2.1, 2.2, 2.3(b) are all multiple-record documents because they contain similar descriptions of several different items. Figure 2.4 shows a car ad linked from *Honda Accord EX* in Figure 2.2, which we call a single-record document because it declares the various features of only one item—the *Honda Accord EX* for sale. In addition to single- and multiple-record documents, Figure 2.3(a) is an application form. When considering a form, we may have, in addition to the labeled form fields, (1) selection lists with possible values of interests, and (2) the results returned, if we can automatically fill in and submit the form using default values as discussed in [45, 75].

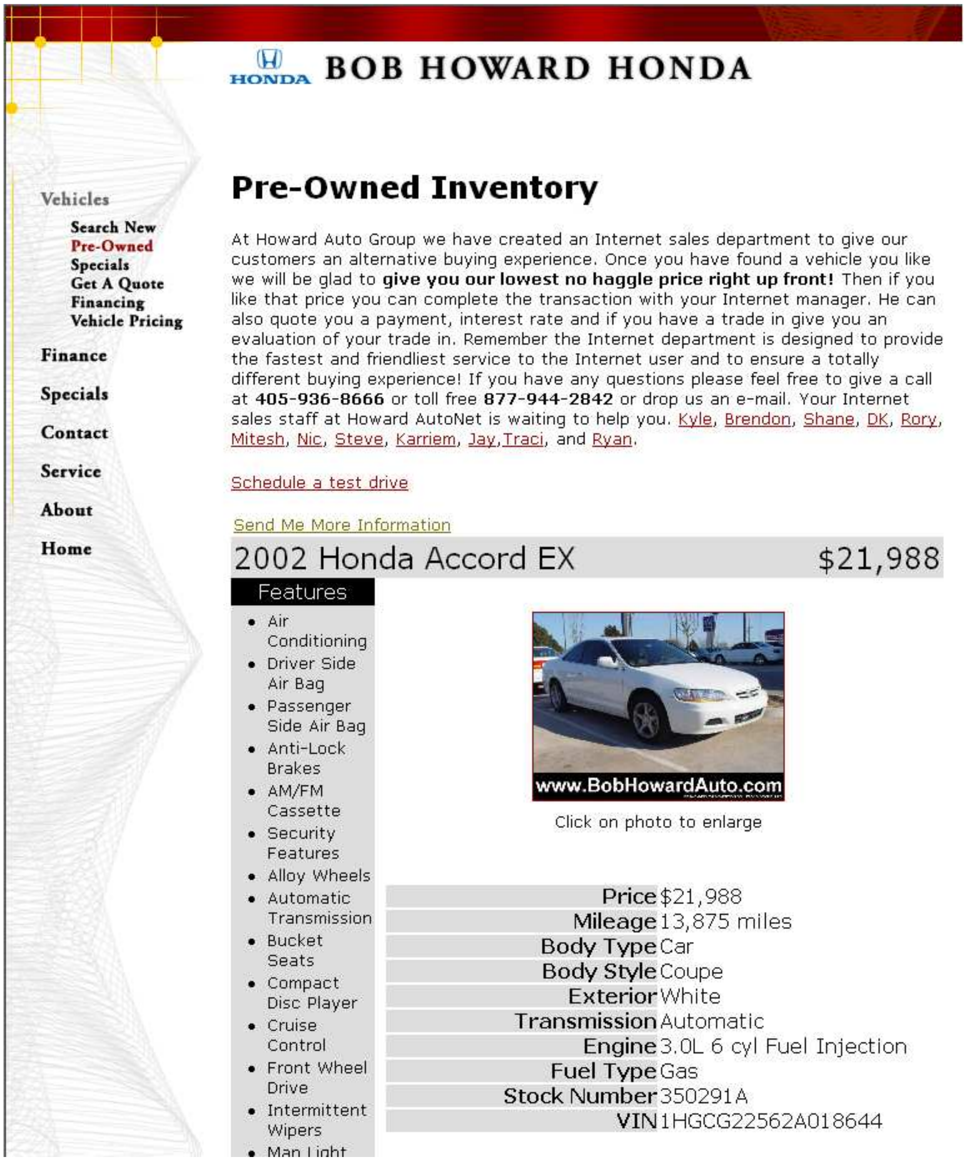


Figure 2.4: Linked page with additional information [7]

```
1. Car [-> object];
2. Car [0:0.975:1] has Year [1:*];
3. Car [0:0.925:1] has Make [1:*];
4. Car [0:0.908:1] has Model [1:*];
5. Car [0:0.45:1] has Mileage [1:*];
6. Car [0:2.1:*] has Feature [1:*];
7. Car [0:0.8:1] has Price [1:*];
8. PhoneNr [1:*] is for Car [0:1.15:*];
9. Year matches [4]
10.         constant {extract “\d{2}”;
11.                 context “\b’[4-9]\d\b”;
12.                 substitute “ˆ” -> ”19”
13.                 ...
14. Mileage matches [8]
15.         ...
16.         keyword “\bmiles\b”, “\bmi\.”, “\bmi\b”,
17.                 “\bmileage\b”;
18. ...
```

Figure 2.5: Car-ads application ontology (partial)

2.2.2 Application Ontologies

We define an application ontology to be a conceptual-model instance that describes a real-world application in a narrow, data-rich domain of interest. Each of our application ontologies consists of two components: (1) an *object/relationship-model instance*, which describes sets of objects, sets of relationships among objects, and constraints over object and relationship sets, and (2) for each object set, a *data frame*, which defines the potential contents of the object set. A data frame for an object set defines the lexical appearance of constant objects for the object set and establishes appropriate keywords that are likely to appear in a document when objects in the object set are mentioned. Figure 2.5 shows part of our car-ads application ontology, including object and relationship sets and cardinality constraints (lines 1-8) and a few lines of the data frames (lines 9-18). The full ontology for car ads is about 600 lines long. Our obituary ontology, which is the other application ontology we discuss in this chapter is about 500 lines long, but it references both a first-name lexicon and a last-name lexicon, which each contain several thousand names.

An object set in an application ontology represents a set of objects which may either be lexical or nonlexical. Data frames with declarations for constants that can potentially populate the object set represent lexical object sets, and data frames without constant declarations represent nonlexical object sets. *Year* (Line 9) and *Mileage* (Line 14) are lexical object sets whose character representations have a maximum length of 4 and 8 characters respectively. *Make*, *Model*, *Price*, *Feature*, and *PhoneNr* are the remaining lexical object sets in our car-ads application; *Car* is the only nonlexical object set.

We describe the constant lexical objects and the keywords for an object set by regular expressions using the Perl syntax. When applied to a textual document, the **extract** clause in a data frame causes a string matching a regular expression to be extracted, but only if the **context** clause also matches the string and its surrounding characters. A **substitute** clause lets us alter the extracted string before we store it in an intermediate file, in which we also store the string's position in the document and its associated object set name. One of the nonlexical object sets is designated as the *object set of interest*—*Car* for the car-ads ontology. The notation "[-> object]" in Line 1 designates the object set of interest.

We denote a relationship set by a name that includes its object set names (e.g. *Car has Year* and *PhoneNr is for Car*). The *min:max* pairs and *min:ave:max* triples in the relationship-set name are *participation constraints*: *min* designates the minimum number of times an object in the object set can participate in the relationship set; *ave* designates the average number of times an object is expected to participate in the relationship set; and *max* designates the maximum number of times an object can participate, with * designating an unknown maximum number of times. The participation constraint on *Car* for *Car has Feature*, for example, specifies that a car need not have any listed features, that a car has 2.1 features on the average, and that there is no specified maximum for the number of features listed for a car.

For our car-ads and obituaries application ontologies, we obtained participation constraints as follows. To make our constraints broadly representative, we selected ten different regions covering the United States and found one car-ads page and one obituary page from each of these regions. From each of these pages we selected twelve individual car-ads/obituaries by taking every $n/12$-th car-ad/obituary, where n was the total number of car-ads/obituaries on the page. We then simply counted by hand and obtained minimum, average, and maximum values for each object set in each relationship set and normalized the values for a single car ad or obituary.

2.2.3 Classification Task

Real-world applications require that the recognition of document applicability be flexible, robust, and scalable. Even if we can manually construct rules to automate this recognition, the automation depends largely on how knowledgable a human expert is, who makes the automatic rules for the particular application domain. Especially when porting to a new application domain, existing recognition rules may no longer be appropriate. To resolve these challenges, in our approach we construct automatic recognition rules via machine learning. We rephrase the problem of recognizing applicable HTML documents for an application into a *classification* task: given a set of HTML documents consisting of semistructured HTML documents, HTML tables, and HTML forms, we attempt to assign each HTML document a concept class, which is either *applicable* or *nonapplicable*, with respect to a particular application. Since we use car ads as a running example in this chapter, we want to classify the multiple-record document in Figure 2.1(a) as being applicable whereas we want to classify the multiple-record document in Figure 2.1(b) as being nonapplicable.

Like a typical classification task, the classification of document applicability proceeds in two phases: *training* and *test*. In the training phase, we supervise the machine to train a *learner* for an application. During the training phase, the learner builds a *classification model*, for example, decision trees [58] for the application. In the test phase, given an HTML document, the learner applies the classification model to predict whether the document is applicable to the application.

2.3 The Approach

In Figure 2.6, we present a high-level architecture of our approach for the applicability checking of HTML documents for an application.

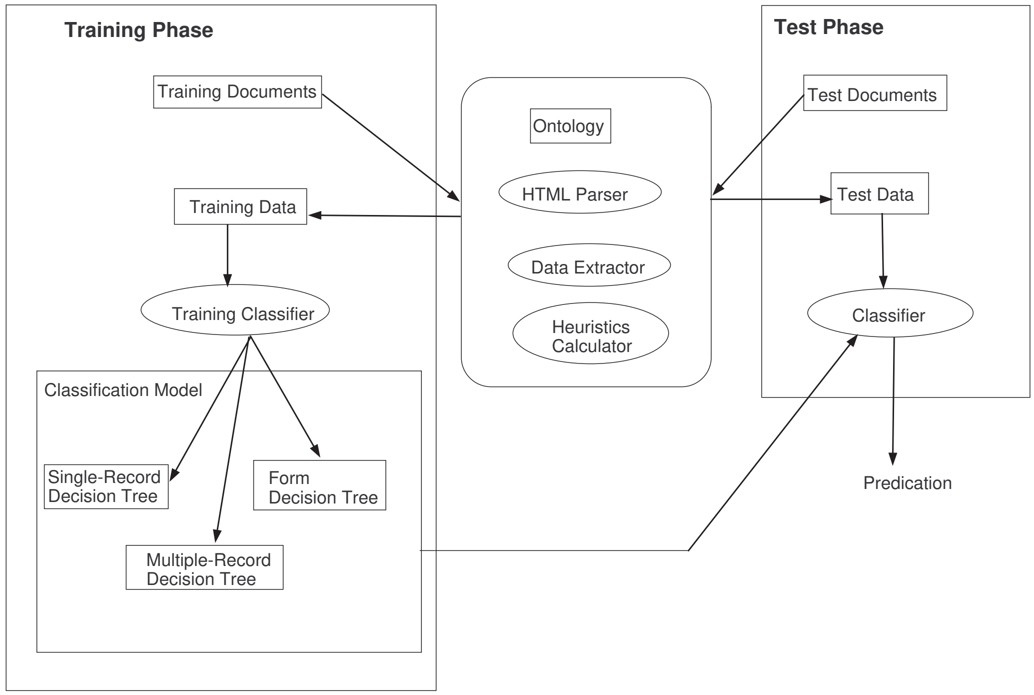


Figure 2.6: High-level architecture for recognizing applicable HTML documents for an application

In the architecture, one application **Ontology**, which specifies object and relationship sets and data frames for an application, is predefined independently of HTML documents. Given an HTML document, we use an **HTML Parser** to parse the document. For each document, we collect two kinds of texts. One kind is the text that appears in the whole document, which we call the *document text*. The other kind are text fragments that appear within individual forms in the document, each of which we call a *form text*. A form text includes the text that labels form fields and values in selection lists. If the document does not contain any form, the set of form texts is empty. Note that the document text of an HTML document subsumes all the form texts that appear within the forms as well as the text that is outside the forms in the document. The **Data Extractor** applies the application ontology using ontology-based data-extraction techniques [21] to retrieve relevant data from all the document and form texts in the document.

Based on the data extracted from a text, which is either a document text or a form text, we construct a list of heuristics to evaluate the applicability of the text to the application ontology by a **Heuristics Calculator**. Each heuristic evaluates the applicability of a document to the application ontology. We normalize a measure for each individual heuristic as a confidence measure in the range from 0 to 1. The higher the confidence value, the more confidently we consider the text appropriate for the application ontology for the particular heuristic. As will become evident, in our approach, we construct heuristics that are tightly dependent on the specification of application ontologies in a flexible and robust way.

We formalize the evaluation of HTML-document applicability to an application as follows. An application ontology O specifies the application. HTML documents are structured objects: a document comprises a document text and a set of form texts. More precisely, a *document* d is a sequence of texts, written $d = [t_d, t_{f_1}, ..., t_{f_n}]$, where n is the number of forms in the document d. Note that $d = [t_d]$ if the document d does not contain any form. Given the application ontology O and an HTML document $d = [t_d, t_{f_1}, ..., t_{f_n}]$, we use m heuristic rules to compute m confidence measures $H_{t_d} = (h_1, h_2, ..., h_m)$ for the document text t_d and use the same m heuristic rules to compute $H_{t_{f_i}} = (h_{i1}, h_{i2}, ..., h_{im})$ for each form text t_{f_i} ($1 \leq i \leq n$). Thus we describe the similarity between the HTML document d and the application ontology O as a heuristic vector $d_H =< H_{t_d}, H_{t_{f_1}}, ..., H_{t_{fn}} >$ over n+1 m tuples of confidence measures. Since we rephrase the recognition of document applicability into a classification problem, we attempt to assign d_H to either a concept class c_P, which represents positive (applicable to the application), or c_N, which represents negative (nonapplicable to the application).

2.3.1 Training Phase

In the training phase, we train a learner using a **Training Classifier**, which for our work is the popular machine learning algorithm C4.5 [58]. C4.5 is a rule post-pruning decision-tree algorithm. The learning task is to check the suitability of documents for a given application ontology (i.e. to do binary classification by returning "Y" (yes) when a document is suitable and returning "N" (no) otherwise). The bias of C4.5 favors the shortest rule, so that if several rules are equally accurate, a decision tree with the fewest branches is chosen. Actually, the training classifier could use other learning algorithms. Indeed, our research group has tried both multivariate statistical analysis [68] and logistic regression [73] as alternative learning algorithms.

With the consideration of three patterns (multiple-record documents, single-record documents, and application forms) distinguished based on the contents of applicable HTML documents with respect to applications, we divide the learning task into three subtasks: (1) suitability of a document text that describes multiple records for one application ontology, (2) suitability of a document text that represents an individual singleton record for one application ontology, and (3) suitability of a form that yields information for one application ontology. We use C4.5 to learn a decision tree for each of the three subtasks.

We employ supervised learning in our approach to train the learner. For each application, a human expert selects a set of HTML documents for the application ontology as **Training Documents**, which we denote as T_O. The human expert selects training documents for each subtask considering the different kinds of HTML documents in the real world. For example, for the car-ads application, we selected semistructured HTML documents as well as HTML tables containing multiple car ads as training documents for the subtask to train the learner so that the learner can obtain the knowledge it needs to classify a multiple-record car-ads document. Various design patterns of HTML tables, including the page in Figure 2.2 for example, can be considered as training documents.

By using the HTML parser, the data extractor, and the heuristics calculator, the learner obtains a heuristic vector $d_H =< H_{t_d}, H_{t_{f_1}}, ..., H_{t_{f_n}} >$ for each training document d. With the collection of heuristic vectors obtained from training documents, the human expert helps the learner construct the **Training Data** for the three subtasks. Each unit in the training data is a list of heuristics associated with a concept class, which we denote as $u = (H_x, c_y)$ where H_x is a list of heuristics in d_H obtained from either a document text or a form text within d and c_y is either c_P or c_N. We call a unit in the set of training data a *training example*. If the concept class c_y in a training example u equals c_P, the training example u is positive, applicable to the application ontology O. Otherwise, the training example u is negative, nonapplicable to O.

The training data contains three groups of training examples, each of which is for one of the three subtasks. For a training document d in the set of training documents

T_O, if it contains a form f_i applicable to O, the human expert uses the list of heuristics $H_{t_{f_i}}$, which is obtained from the form text t_{f_i}, to construct a positive training example $(H_{t_{f_i}},\ c_P)$ for the subtask specifying the applicability of a form to the application ontology O. Otherwise if the form f_i is nonapplicable to O, the human experts builds a negative training example $(H_{t_{f_i}},\ c_N)$ for the subtask. If the document is a single-record document applicable to O, the human expert uses the list of heuristics H_{t_d}, which is obtained based on the document text t_d, to build a positive training example $(H_{t_d},\ c_P)$ for the subtask specifying the applicability of a single-record document applicable to O, or vice versa a negative training example. Similarly, the human expert uses H_{t_d} obtained from the document text of an HTML document to build a training example for the subtask specifying the applicability of a multiple-record document applicable to O.

The C4.5 algorithm knows how the heuristics in the training examples should be optionally combined to best match application ontologies with texts that appear in documents. Thus the learner builds a **Classification Model**, which we denote as M_O, as the output of the training phase. The classification model contains three decision trees. One tree τ_1 is a set of rules to decide if a document is a single-record document applicable to an application ontology, which we call a *single-record tree*. The second decision tree τ_2 is a set of rules to evaluate if a document is a multiple-record document applicable to an application ontology, which we call a *multiple-record tree*. The last decision tree τ_3 is a set of rules to evaluate if an HTML form is applicable to an application ontology, which we call an *application-form tree*.

2.3.2 Test Phase

In the test phase, we use a set of HTML documents, which we call **Test Documents** and denote them as T'_O, to evaluate the performance of the learner trained in the training phase for the application ontology O. Given the classification model built in the training phase, we use an algorithm in Figure 2.7, which is the **Classifier** to test the applicability of an HTML document. The input to the test algorithm is the classification model and an HTML document d from the set of test documents. The output is a prediction about the applicability of d to O. Since the classification model is composed of three trees to classify

Input: an HTML document d, an application ontology O,
  and a classification model M_O (τ_1, τ_2, τ_3).
Output: prediction
  Build d_H = <H_{t_d}, $H_{t_{f_1}}$, $H_{t_{f_2}}$, ..., $H_{t_{f_n}}$> for d
  //check the document text in d
  call EvaluateDocText(d, H_{t_d}, M_O)
  //check the form texts in d
  for each $H_{t_{f_i}}$ in d_H where $1 \leq i \leq$ n
    call EvaluateFormText(d, $H_{t_{fi}}$, M_O)
  Output the prediction for d

Figure 2.7: Test algorithm to recognize applicable HTML documents

a single- or multiple-record document or an application form to an application ontology, a positive prediction results for a test document if the learner classifies the document as positive based on any one of the three decision trees. Figure 2.8 shows two subprocedures of the test algorithm that check the applicability based on the evaluation over either a document text or a form text. In Figure 2.6, the **Test Data** consists of the heuristic vectors computed in the algorithm for the test documents, and the **Applicability Predictions** are the predictions output from the test algorithm for the test documents.

In the algorithm of Figure 2.7, the learner classifies an HTML document by exploiting the two kinds of texts that appear in the document based on the classification model M_O. In addition to the texts within the document, the learner can further check available auxiliary information such as linked pages to further re-evaluate the document. For example, the HTML table in Figure 2.2 contains several links leading to the detailed descriptions of car ads in the document. Figure 2.4, which is a linked page from the top page in Figure 2.2 describes a car ad in detail. If we can evaluate the linked page, for example, the document in Figure 2.4 as applicable to the application, we can use this information to help the classification of the top page in Figure 2.2. Intuitively, if a top page leads to multiple linked pages which are applicable documents, we have more confidence that the top page contains multiple records that are of interest. Figure 2.9 shows the algorithm to evaluate the document text of an HTML document by exploiting applicable documents in linked pages. The algorithm does not exploit all linked pages from the top-page HTML document

sub EvaluateDocText(d, H_{t_d}, M_O)
    //check a document text
    Evaluate H_{t_d} using τ_1 and obtain c_1
    Evaluate H_{t_d} using τ_2 and obtain c_2
    if either c_1 or c_2 equals c_P
        Output c_P for the document text of d
    else
        Output c_N for the document text of d
end EvaluateDocText

sub EvaluateFormText(d, $H_{t_{f_i}}$, M_O)
    //check a form text
    Evaluate $H_{t_{f_i}}$ using τ_3 in M_O and obtain c
    Output c for the form text of f_i
end EvaluateFormText

Figure 2.8: Subprocedures of prediction algorithm

Input: an HTML document d, an application ontology O,
    a sample size N, and a classification model M_O (τ_1, τ_2, τ_3).
Output: prediction
    call LocateUsefulLinks(d, M_O, N) and output L
    Randomly select subsequent links L_S from L
    for each linked page d' in L_S
        Compute $H_{t_{d'}}$ over the document text of d'
        call EvaluateDocText(d', $H_{t_{d'}}$, M_O) and output c'
        if c' equals c_P
            Retrieve text from d'
            Insert the text in d
    Compute H_{t_d} over the document text t_d of d
    if t_d in d is modified
        Evaluate H_{t_d} using τ_2 and output c
    else
        call EvaluateDocText(d, H_{t_d}, M_O) and output c
    Output c for the document text of d

Figure 2.9: Evaluation of the document text of an HTML document by applying applicable linked pages

```
sub LocateUsefuleLinks(d, M_O, N)
    Collect all the links L in d
    Group L into subgroups by link-address prefixes
    Sort the subgroups in descending order by sizes
    for each G in the subgroups
        Sample and evaluate N links in G based on M_O
        If there is an applicable linked document
            Output G and break the loop
    if no subgroup is found
        Output an empty set
end LocateUsefulLinks
```

Figure 2.10: Subprocedure to locate useful links in an HTML document

d. Instead, we first call subprocedure in Figure 2.10 to locate a group of potential useful links from d. In the procedure to locate the useful links, we sort the subgroups of links in descending order based on the number of links in each subgroup since the number of the links that are of interest usually is the largest in an applicable document. One heuristic we used is that the links that lead to applicable documents in d are grouped together. In the loop of the algorithm in Figure 2.10, if the number of the links in a subgroup is less than the sample size N, we ignore the evaluation of the particular subgroup. Thus, if there are some randomly referenced links in the page, the algorithm ignores the links if we select an appropriate sample size N. After locating useful links, the algorithm in Figure 2.10 checks linked pages to see if they are individually applicable. If so, the algorithm checks the applicability of the organized document with the text of the linked pages inserted. The basic tradeoff in response time performance between the procedure $EvaluateDocText$ in Figure 2.8 and the algorithm in Figure 2.9 is the time to evaluate the document text in an HTML document versus the time to locate applicable link pages in the document and evaluate both the document text and linked pages for the document.

The other kind of auxiliary information we can use depends on the methods [45, 75] of filling form automatically. Figure 2.11 shows the algorithm to further re-evaluate an HTML form by considering a document retrieved by form-filling. We can exploit the

Input: an HTML form f_i, an application ontology O,
and a classification model M_O (τ_1, τ_2, τ_3).
Output: prediction
Retrieve a document d' by filling in and submit f_i
Compute $H_{t_{d'}}$ over the document text of d'
call EvaluateDocText(d', $H_{t_{d'}}$, M_O) and output c
Output c for the form f_i

Figure 2.11: Evaluation of an HTML form using a retrieved document obtained by applying the form-filling methods

retrieved documents using two strategies based on the preference of the system users: if the system users prefer a better recall ratio, we re-evaluate a test document d using the algorithm in Figure 2.11 when the learner classifies d as a nonapplicable document to the application ontology using the algorithm in Figure 2.7; otherwise, if the system users prefer a better precision ratio, the learner checks the available retrieved document using the form-filling methods for a test document d if the learner classifies d as a positive application form using the test algorithm in Figure 2.7. The basic tradeoff in response time performance between the procedure $EvaluateFormText$ in Figure 2.8 and the algorithm in Figure 2.11 is the time to evaluate the form text that appears in an HTML form versus the time to fill in and submit the HTML form and evaluate the returned document.

Note that C4.5 trains the learner to obtain the classification model by applying three sets of training examples independently in the training phase. Thus, in the test phase, it is possible that the learner classifies a test document as both a single-record document and a multiple-record document applicable to the application ontology based on the document text in the document. Moreover, it also could predict one applicable document contains both applicable forms as well as a singleton or multiple records for the application. In our approach, however, in the test phase, we are only interested in a prediction. Thus, we declare a document if any one of the three trees in the classification model returns c_P.

2.4 Recognition Heuristics

In the high-level architecture of our approach in Figure 2.6, the heuristics calculator computes heuristic measures over document and form texts that appear in a document. In our approach, we consider three kinds of different heuristics: *densities*, *expected values*, and *grouping*. Given an application ontology O, the set of density heuristics measure the densities of constants and keywords defined in the application ontology O that appear in a text t^d, which is either a document text or a form text. The expected-values heuristic uses the Vector Space Model (VSM) [63], a common information-retrieval measure of document relevance, to compare the number of constants expected for each object set, as declared in O, to the number of constants found in t^d for each object set. The grouping heuristic measures the occurrence of groups of lexical values found in t^d with respect to expected groupings of lexical values implicitly specified in O.

The next three subsections define these heuristics, explain the details about how we provide a measure for each heuristic, and give examples to show how they work. When reading these subsections, bear in mind that in creating these heuristics, we favored simplicity. More sophisticated measures can be obtained. For example, for density measures we could account for uncertainty in constant and keyword matches [22]. For expected values, we could more accurately match object sets with recognized values by using our more sophisticated downstream heuristics [21, 27]. For grouping, we could first compute record boundaries [24] and appropriately rearrange record values [27]. However, more sophisticated measures are more costly. We have chosen to experiment with less costly heuristics, and, as will be shown, our results bear out the seeming correctness of this choice.

2.4.1 Density Heuristics

A text t^d parsed from an HTML document d that is relevant to a particular application ontology O should include many constants and keywords for object sets defined in the ontology. Based on this observation, we define a set of density heuristics. We compute the density heuristics with respect to an application ontology and each object set that has keywords or constants specified in the ontology. We compute the density of t^d with respect to an application ontology O as follows:

$$Density(t^d, O) = \text{total number of matched characters / total number of characters}$$

where *total number of matched characters* is the number of characters of the constants and keywords recognized by O in t^d, and *total number of characters* is the total number of characters in t^d. Further, we compute the density of t^d with respect to an object set o in O as follows:

$$Density(t^d, o) = \text{total number of matched characters for } o \text{ / total number of characters}$$

where *total number of matched characters for o* is the number of characters of the constants and keywords recognized by regular expressions specified for o in t^d, and *total number of characters* is the same as in the computation of $Density(t^d, O)$.

We must be careful, of course, not to count characters more than once. For example, in the phrase "asking only 18K," a car-ads application ontology might recognize "18K" as potentially both a price and a mileage. Nevertheless, we should only count the number of characters as three, not six. Further, we need determine whether we count the value "18K" for a price or for a mileage.

Consider the document text $t_d^{d_a}$ in the multiple-record document d_a in Figure 2.1(a). Recall that the nonlexical object set of the car-ads application ontology is Car, and the lexical object sets are $Year$, $Make$, $Model$, $Mileage$, $Price$, $Feature$, and $PhoneNr$. Some of the lexical values found in t^{d_a} include 1989 ($Year$), \$1900 ($Price$), 100K ($Mileage$), Auto ($Feature$), Cruise ($Feature$), (336)835-8579 ($PhoneNr$), Subaru ($Make$), and SW ($Model$). The keywords "Cars for Sale" for the object set of interest Car, "miles" and "mileage" for $Mileage$, and "Call" for $PhoneNr$ appear in d_a. The total Number of characters in $t_d^{d_a}$ is 1992, whereas the number of matched characters is 696. Hence, the $Density(t_d^{d_a}, O)$ is 0.3493 = 696/1992. For each object set in the car-ads application ontology O, there is also a density measure. For example, the number of matched characters for $Make$ is 47. Therefore, $Density(t_d^{d_a}, Make)$ is 0.0236 = 47/1992.

When we apply the density heuristics for the car-ads application ontology to the document text $t_d^{d_b}$ of the document d_b in Figure 2.1(b), the densities are much lower. Although no makes, models, or car features appear, there are years, prices, and phone numbers and the ontology (mistakenly) recognizes "10,000" (in "10,000 SQ. FT.") and

"401K" (the retirement plan) as potential mileages. Altogether, 229 characters of 2627 are recognized by the car-ads ontology. Thus the density to the car-ads application ontology $Density(t_d^{d_b}, O)$ is 0.0871. There are also eight other densities, one for each object set. For example, the document text of d_b contains keywords and values for $PhoneNr$, and the density for $PhoneNr$ is 0.0533. The density for Car is 0.0 since the document text does not contain any keywords for the object set of interest, Car, in the car-ads application ontology.

2.4.2 Expected-Values Heuristic

We apply the VSM model to measure whether a text t^d parsed from an HTML document d has the number of values expected for each lexical object set of an application ontology O. Based on the lexical object sets and the participation constraints in O, we construct an ontology vector V_O. Based on the same lexical object sets and the number of constants recognized for these object sets by O in t^d, we construct a document vector V_{t^d}. We measure the relevance of t^d to O with respect to our expected-values heuristic by observing the cosine of the angle between V_{t^d} and V_O.

To construct the ontology vector V_O, we (1) identify the lexical object-set names (these become the names of the coefficients of V_O) and (2) determine the average participation (i.e. the expected frequency of occurrence) for each lexical object set with respect to the object set of interest specified in O (these become the values of the coefficients of V_O). Since we do not in general know (and indeed do not care) how many records we will find in documents given to us, we normalize these values for a single record. For example, the ontology vector for the car-ads application ontology is $<$ $Year$:0.975, $Make$:0.925, $Model$:0.908, $Mileage$:0.45, $Price$:0.8, $Feature$:2.1, $PhoneNr$:1.15 $>$, where these values are obtained as explained in Section 2.2.2. Thus, for a typical single car ad we would expect almost always to find a year, make, and model, but we only expect to find the mileage about 45% of the time, the price about 80% of the time. Further, we expect to see a list of features that on the average have a couple of items in it, and we expect to see a phone number and sometimes more than one phone number.[1]

The names of the coefficients of V_{t^d} are the same as the names of the coefficients of V_O. We obtain the value of each coefficient of V_{t^d} by automatically counting the number of appearances of constant values in t^d that belong to each lexical object set. Table 2.1 shows the values of the coefficients of the document vector for the document text of the car-ads document in Figure 2.1(a), and Table 2.2 shows the values of the coefficients of the document vector for the document text of the non-car-ads document in Figure 2.1(b). Observe that for document vectors we use the actual number of constants found in a document. To get the average (normalized for a single record), we would have to divide by the number of records—a number we do not know with certainty. We can estimate the number of records by dividing the length of the document vector by the length of the ontology vector. Indeed, we use this information downstream, but here we are still trying to determine whether the given document applies to the ontology. If it does, the length-division estimate for the number of records makes sense; otherwise, the estimate may be nonsense. Therefore, we do not normalize, but instead merely compare the cosine of the angles between the vectors to get a measure for our expected-values heuristic.

We have discussed the creation of a document vector as though correctly detecting and classifying the lexical values in a text in a document was easy—but it was not easy. We identify potential lexical values for an object set as explained in Section 2.2.2; this can be error-prone, but we can adjust the regular expressions to improve this initial identification and achieve good results [21]. After initial identification, we must decide which of these potential object-set/constant pairs to accept. In our downstream processes, we use sophisticated heuristics based on keyword proximity, application-ontology cardinalities, record boundaries, and missing-value defaults to best match object sets with potential constants. For upstream ontology/document matching we use techniques that are far less sophisticated and thus also far less costly. In our simple upstream procedures we consider only three cases: (1) a recognized string appears without required accompanying keywords,

[1]It is easy to see that the variance might be useful, as well, but we found that the expected numbers were sufficient to get good results for the examples we tried.

Name of Lexical Object Set	Corresponding Lexical Values Found in the Document	Number of Lexical Values
Year	1989, 1998, 1994, 1999, ’53, 1973, 1977, 95, 1996, . . .	16
Make	Subaru, HONDA, Chevy, Olds, FORD, VW, Buick, Mercury, . . .	10
Model	SW, Elantra, ACCORD, GRAND AM, Cutlass, CONTOUR, JETTA, . . .	12
Mileage	100K, 27000, 26000, 45K, 34K, 109000	6
Price	$1900, $14,000, $8500, $4500, $5300, $11,000, $6995, $4995, $1880, . . .	11
Feature	Auto, Black, 4 door, pb, ps, cruise, am/fm, cassette, stereo, green, . . .	29
PhoneNr	(336)835-8579, (336)526-5444, (336)526-1081, (336)366-4996, . . .	15

Table 2.1: Lexical values found in the multiple-record car advertisements in Figure 2.1(a)

(2) a recognized string has no overlap either partially or completely with any other recognized string, and (3) a recognized string does overlap in some way with at least one other recognized string. If one object set o is a specialization of another object set described in an application ontology, the keywords specified for o in the application ontology are mandatory. For example, there are several dates in the obituaries application: $Death\ Date$, $Birth\ Date$, $Funeral\ Date$, $View\ Date$, and $Interment\ Date$. All dates are specializations of $Date$ in the obituary application ontology. To distinguish the multiple kinds of dates in HTML documents, it is necessary to use keywords to sort out the difference. For Case 1, we reject a recognized string if mandatory keywords are not present. For Case 2, we accept the recognized string for an object set even if the sophisticated downstream processes would reject it. For Case 3, we resolve the overlap simplistically, as follows. There are three subcases: (1) exact match, (2) subsumption, and (3) partial overlap. (1) If a lexical value v is recognized as potentially belonging to more than one lexical object set, we use the closest keyword that appears before or after v to determine which object set to choose; if no applicable keyword is found, we choose one of the object sets arbitrarily. (2) If a lexical value v is a proper substring of lexical value w, we retain w and discard v. (3)

Name of Lexical Object Set	Corresponding Lexical Values Found in the Document	Number of Lexical Values
Year	1999, 1998, 60, 401K, 50, 80	6
Make		0
Model		0
Mileage	10,000, 401K	2
Price	$17,500, $10,971, $27,850, $19,990, $79,850, $42,990, $129,650, $78,850	8
Feature		0
PhoneNr	281-2051, 281-4060, 218-281-1128, 281-3631, 281-3872, 218-281-5113, 218-281-5113, 800-532-7655, 281-1970, 800-406-5126, 281-1128	11

Table 2.2: Lexical values found in the multiple-record *Items for Sale* document in Figure 2.1(b)

If lexical value v and lexical value w appear in a Web document, such that a suffix of v is a prefix of w, we retain v and discard w.

As mentioned, we measure the similarity between an ontology vector V_O and a document vector V_{t^d} by measuring the cosine of the angle between them. In particular, we use the *Similarity Cosine Function* defined in [63], which calculates the acute angle $Similarity(t^d, O)$ = cos θ = P/N, where P is the inner product of the two vectors, and N is the product of the lengths of the two vectors. When the distribution of values among the object sets in V_{t^d} closely matches the expected distribution specified in V_O, the angle θ will be close to zero, and cos θ will be close to one.

Consider the car-ads application ontology O as shown in Figure 2.5 and the Web document d_a as shown in Figure 2.1(a). The coefficients of V_O for O are 0.975, 0.925, 0.908, 0.45, 0.8, 2.1, and 1.15, which are the expected frequency values of lexical object sets $Year$, $Make$, $Model$, $Mileage$, $Price$, $Feature$, and $PhoneNr$, respectively for a single ad in the car-ads application ontology. The coefficients of $V_{t_d^{d_a}}$ for d_a are 16, 10, 12, 6, 11, 29, and 15 (see the last column of Table 2.1), which are the actual number of appearances of the lexical values in d_a. We thus compute $Similarity(t_d^{d_a}, O)$ to be 0.9956. Now consider the car-ads application ontology O again and the Web document d_b as shown in Figure 2.1(b). The coefficients of V_O are always the same, but the coefficients of $V_{t_d^{d_b}}$

for d_b are 6, 0, 0, 2, 8, 0, and 11 (see the last column of Table 2.2). We thus compute $Similarity(t_d^{d_b}, O)$ to be 0.5669.

2.4.3 Grouping Heuristic

A text t^d of an HTML document d likely applies to an application ontology if the values in the text form groups that can be recognized as records for O. As a simple heuristic to determine whether the recognized values are interleaved in a way that could be considered consistent with potential records of O, we consider the sequence of values in a document that should appear at most once in each record and measure how well they are grouped.

We refer to an object set whose values should appear at most once in a record as a *1-max lexical object set*. Maximum participation constraints in an ontology constrain the values of the 1-max object sets to appear at most once in a record. For example, in the car-ads application ontology, the 1-maximum on *Car* in the relationship set *Car [0:0.975:1] has Year [1:*]* specifies that $Year$ is a 1-max object set. Other 1-max lexical objects in the car-ads ontology are $Make$, $Model$, $Mileage$, and $Price$.

Instead of counting the number of 1-max lexical objects in an application ontology O, a more adequate counting approach is to sum the average values expected for the 1-max objects in O. Since the average values expected for $Year$, $Make$, $Model$, $Mileage$, and $Price$ in the car-ads ontology are 0.975, 0.925, 0.908, 0.45, and 0.8, respectively, the anticipated number of lexical values from these object sets in a car advertisement is 4.058. To obtain an expected group size, we truncate the decimal value of the sum.

The expected group size n is an estimate of the number of 1-max object-set values we should encounter in a document within a single record. On the average, each record should have n 1-max object sets. Thus, if we list all recognized 1-max object-set values in the order they occur in a document d and divide this sequence into groups of n, each group should have n values from n different object sets. The closer a document comes to this expectation, the better the grouping measure should be. For the multiple-record car-ads Web document in Figure 2.1(a), Figure 2.12(a) shows the first four groups of 1-max lexical object-set values extracted from the document. Similarly, Figure 2.12(b)

Year: 2000
Year: 1989
Make: Subaru
Model: SW
-- Nr of Distinct "One Max" Object Sets: 3

Price: 1900
Year: 1998
Model: Elantra
Year: 1994
-- Nr of Distinct "One Max" Object Sets: 3

Make: HONDA
Model: ACCORD
Mileage: 100000
Year: 1999
-- Nr of Distinct "One Max" Object Sets: 4

Model: GRAND AM
Mileage: 27000
Price: 14000
Year: 1953
-- Nr of Distinct "One Max" Object Sets: 4

(a) First four groups of 1-max lexical values extracted from Figure 2.1(a)

Year: 1999
Year: 1998
Year: 1960
Mileage: 10000
-- Nr of Distinct "One Max" Object Sets: 2

Mileage: 401000
Year: 1940
Price: 17500
Price: 10971
-- Nr of Distinct "One Max" Object Sets: 3

Year: 1950
Price: 27850
Price: 19990
Year: 1980
-- Nr of Distinct "One Max" Object Sets: 2

Price: 79850
Price: 42990
Price: 129650
Price: 78850
-- Nr of Distinct "One Max" Object Sets: 1

(b) First four groups of 1-max lexical values extracted from Figure 2.1(b)

Figure 2.12: Groups of 1-max lexical values extracted from advertisement HTML documents

shows the first four groups of 1-max lexical object-set values extracted from the document in Figure 2.1(b).

We measure how well the groups match the expectations with a grouping factor (denoted $Grouping$), which is calculated as follows:

$$Grouping(t^d, O) = \frac{\text{Sum of Distinct Lexical Values in Each Group}}{\text{Number of Groups} \times \text{Expected Number of Values in a Group}}$$

For example, the number of extracted groups from the document text $t_d^{d_a}$ of d_a in Figure 2.1(a) is 13 (1 group of 2, 5 groups of 3, and 7 groups of 4). Since the number of anticipated lexical values in each group is four, $Grouping$ of $t_d^{d_a}$ is

$$Grouping(t_d^{d_a}, O) = \frac{(2 \times 1) + (3 \times 5) + (4 \times 7)}{(1 + 5 + 7) \times 4} = 0.8653$$

By way of comparison, the number of extracted groups from the document text $t_d^{d_b}$ of HTML document d_b in Figure 2.1(b) is 4 (1 group of 1, 2 groups of 2, and 1 group of 3). Since the number of anticipated lexical values in each group is four, $Grouping$ of $t_d^{d_b}$ is 0.5.

Application	*Multiple-Record (semistructured/table/negative)*	*Single-Record postive/negative*	*Form positive/negative*
Car Ads	31/112/363	614/636	50/69
Obituaries	68/0/135	62/135	30/32

Table 2.3: Training data for car ads and obituaries

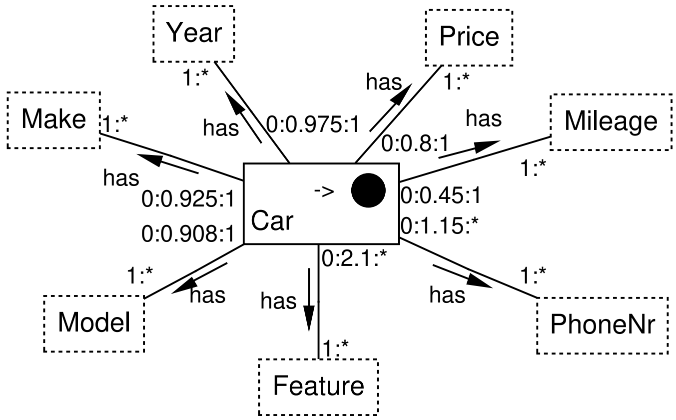
2.5 Empirical Evaluation

We evaluated our approach on two real-world applications: *car ads* and *obituaries*. Our goals were to evaluate the system performance over multiple kinds of HTML documents for real-world applications.

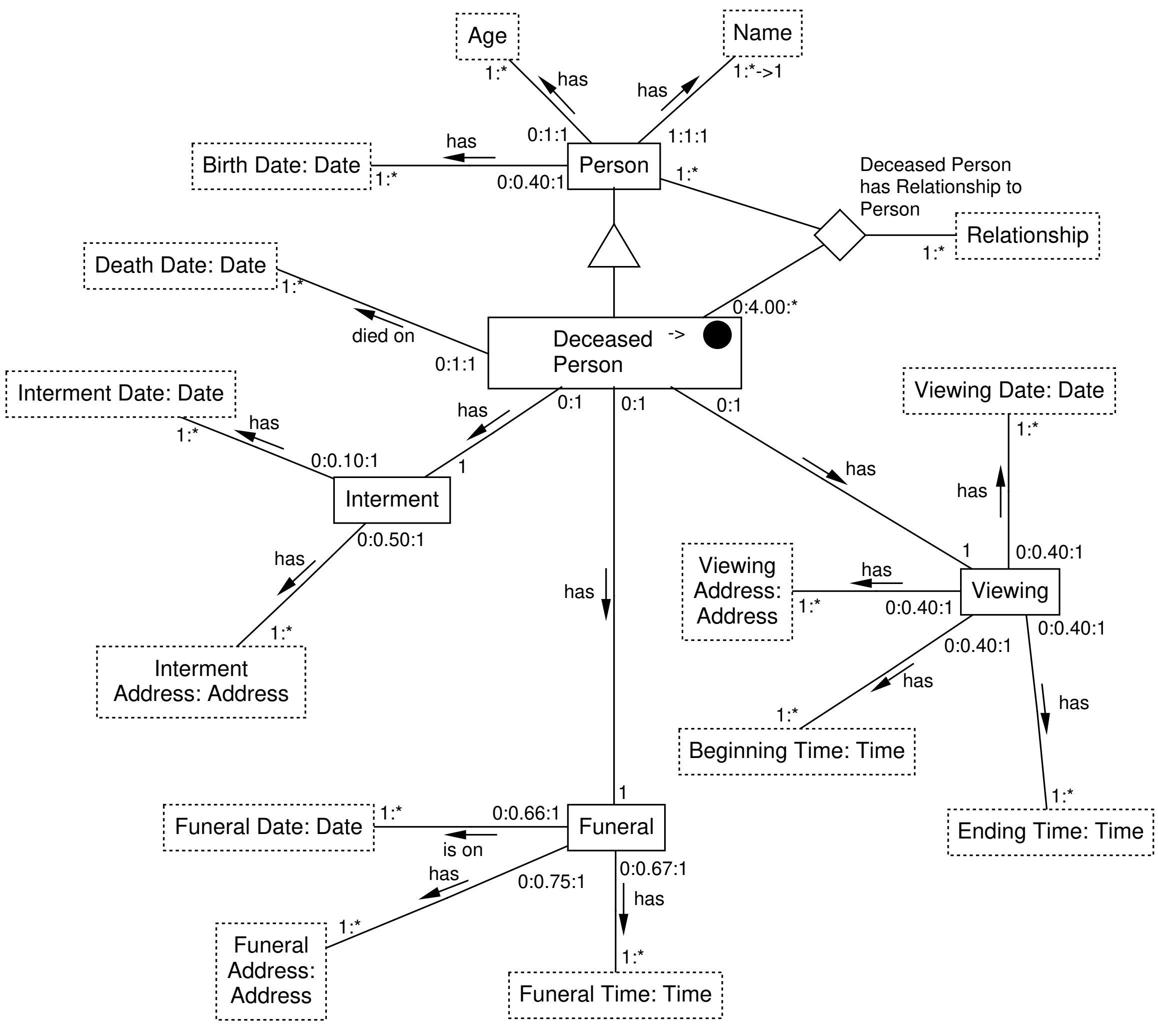
2.5.1 Applications and HTML documents

Figure 2.13 shows the graphical version of the two application ontologies. The car-ads application in Figure 2.13(a) is the running application example we are using in this chapter. It is relatively simpler than the obituaries application in Figure 2.13(b). Not only does the obituaries application has more object sets and relationship sets, but these object and relationship sets are also more complex. One relationship set is ternary, more than one object set is nonlexical, and several relationship sets are specializations—for example, both $Birth\ Date$ and $Death\ Date$ are specializations of $Date$ as denoted by the colon in the name of these object sets.

For the car-ads application, we collected three sets of HTML documents: semistructured HTML documents, HTML tables, and HTML forms. For the obituaries application, we collected only semistructured and HTML form documents. (Obituaries rarely, if ever, appear as tables having attributes such as $Deceased\ Name$, Age, $Death\ Date$, etc.) We divided the documents for each application into two sets: training documents and test documents. Table 2.3 shows the characteristics of the training data for car ads and obituaries obtained from the training documents. For each application, there are three sets of training examples, one each for single-record, multiple-record, and application-form trees.



(a) Car-ads application ontology



(b) Obituary application ontology

Figure 2.13: Graphical versions of application ontologies

Application	*Semistructured*	*HTML Table*	*HTML Form*	*Negative*
Car Ads	20	10	10	40
Obituaries	30	0	10	40

Table 2.4: Test documents for car ads and obituaries

Table 2.4 shows the distributions of semistructured HTML documents, HTML tables, and HTML forms in the test documents. For the car-ads application, 10 of the semistructured HTML documents contain multiple-record car ads, and 10 of them contain single-record car ads. All of the HTML tables contain multiple-record car ads. For the obituaries application, the semistructured HTML documents contain 20 multiple-record obituaries and 10 single-record obituaries. Among the 20 multiple-record obituary documents, 10 documents contain only partial obituaries. The 10 documents lead to linked pages, some of which contain complete obituary records. The 10 HTML form documents for each application contain application forms for the application ontology without single- or multiple-records that are of interest. The negative documents collected for each application contain documents with applications similar to those of application ontologies. For example, we included forms to retrieve used auto parts, car reviews, and motorcycle sales to test the learner trained for car ads, and we included birth and marriage announcements, genealogy records and forms, and bibliographies to test the learner trained for obituaries.

Even though some of the semistructured HTML documents and HTML table documents in Table 2.4 contained additional nonapplicable application forms, we expected that the learners would produce appropriate predictions based on the document texts that appear in the documents rather than the nonapplicable form texts. For the HTML form documents, since they did not contain application records, we expected that the learners would produce the positive predictions using the application-form decision trees based on form texts that appear within forms.

2.5.2 Classification Models

Figure 2.14 and Figure 2.15 respectively show the classification models the learners built for car ads and obituaries. Each classification model contains three decision trees for an application ontology O, which are for multiple-record documents, single-record documents, and application forms. Within one tree, a node denotes a predicate using heuristics measures. For example, "vsm" and "grouping" are two measures for a text t^d computed based on the heuristics calculations of expected-values, $Similarity(t^d,\ O)$, and grouping, $Grouping(t^d,\ O)$. The "density" is a density measure $Density(t^d,\ O)$ of the text t^d with respect to the application ontology O—for either car ads or obituaries. The object-set names "Car" and "Deceased Person" denote the densities of the keywords specified for the object sets of interest in the two application ontologies. Other object-set names denote densities of lexical values and keywords for object sets in the two application ontologies. For example, *Make* denotes a density $Density(t^d,\ Make)$ of the text t^d with respect to the lexical object set *Make* in the application ontology for car ads. The parenthetical numbers (x/y) following "Y" and "N" for a decision-tree leaf L give the total number of training examples x classified for L and the number of incorrect training examples y classified for L.

Given the decision trees for the two applications in Figure 2.14 and Figure 2.15, we can see that the learners used different combinations of heuristics to check the applicability. For both applications, however, the learners largely exploited density heuristics. Figure 2.14 shows that the learner trained for the car-ads application applied only the density heuristics to classify application forms. Especially Figure 2.15 shows that "vsm", the expected-values heuristic, was not useful for the learner of the obituaries application.

2.5.3 Experiments

For each application, we performed two sets of experiments. First, we measured the precision, recall, and the F-measure of our approach, including an investigating of how sensitive the performance is with respect to the analysis of linked pages and the application

```
Make <= 0.12701
|  Car <= 0.006143: N (74.0/7.0)
|  Car > 0.006143
|  |  density <= 0.161905: N (4.0/1.0)
|  |  density > 0.161905: Y (5.0)
Make > 0.12701: Y (36.0)
```

(a) Form decision tree

```
density <= 0.179178
|  vsm <= 0.768438: N (440.0/1.0)
|  vsm > 0.768438
|  |  Mileage <= 0.00666
|  |  |  Year <= 0.009579: N (17.0)
|  |  |  Year > 0.009579
|  |  |  |  Price <= 0.000641: N (2.0)
|  |  |  |  Price > 0.000641: Y (2.0)
|  |  Mileage > 0.00666: Y (3.0)
density > 0.179178
|  grouping <= 0.604167
|  |  Price <= 0.007174: N (5.0/1.0)
|  |  Price > 0.007174: Y (3.0)
|  grouping > 0.604167: Y (134.0)
```

(b) Multiple-record tree

```
Price <= 0.001732
|  Car <= 0.008435: N (545.0/1.0)
|  Car > 0.008435
|  |  Feature <= 0.006432: N (10.0)
|  |  Feature > 0.006432: Y (19.0/3.0)
Price > 0.001732
|  Make <= 0.001559: N (39.0/3.0)
|  Make > 0.001559
|  |  vsm <= 0.598958
|  |  |  PhoneNr <= 0.002147: N (31.0/4.0)
|  |  |  PhoneNr > 0.002147: Y (28.0/3.0)
|  |  vsm > 0.598958
|  |  |  grouping <= 0.611111
|  |  |  |  grouping <= 0.519231: Y (50.0/3.0)
|  |  |  |  grouping > 0.519231: N (10.0/2.0)
|  |  |  grouping > 0.611111: Y (518.0/2.0)
```

(c) Single-record tree

Figure 2.14: Classification model for car ads

```
Funeral Date <= 0.002303
|   Death Date <= 0.022727
|   |   Deceased Person <= 0.009384: N (34.0/4.0)
|   |   Deceased Person > 0.009384: Y (5.0/1.0)
|   Death Date > 0.022727: Y (5.0)
Funeral Date > 0.002303: Y (18.0/1.0)
```

(a) Form decision tree

```
Death Date <= 0.005002
|   Funeral Date <= 0.010844: N (119.0)
|   Funeral Date > 0.010844: Y (9.0/4.0)
Death Date > 0.005002
|   density <= 0.118065: N (7.0)
|   density > 0.118065: Y (68.0/5.0)
```

(b) Multiple-record tree

```
grouping <= 0.277778
|   Relative Name <= 0.047308: N (109.0/1.0)
|   Relative Name > 0.047308
|   |   Death Date <= 0.00202: N (4.0/1.0)
|   |   Death Date > 0.00202: Y (4.0)
grouping > 0.277778
|   Deceased Person <= 0.023715
|   |   Birth Date <= 0.036281
|   |   |   Death Date <= 0.001617
|   |   |   |   Funeral Date <= 0.013723: N (7.0/1.0)
|   |   |   |   Funeral Date > 0.013723: Y (3.0)
|   |   |   Death Date > 0.001617
|   |   |   |   Interment Address <= 0.022565
|   |   |   |   |   density <= 0.070368
|   |   |   |   |   |   Deceased Person <= 0.006863
|   |   |   |   |   |   |   density <= 0.040834: N (3.0/1.0)
|   |   |   |   |   |   |   density > 0.040834: Y (14.0)
|   |   |   |   |   |   Deceased Person > 0.006863: N (3.0)
|   |   |   |   |   density > 0.070368: Y (36.0)
|   |   |   |   Interment Address > 0.022565: N (4.0/1.0)
|   |   Birth Date > 0.036281: N (3.0)
|   Deceased Person > 0.023715: N (7.0)
```

(c) Single-record tree

Figure 2.15: Classification model for obituaries

of the form-filling methods in [45, 75]. Second, we conducted studies to evaluate the contributions of heuristics: *Densities*, *Expected Values*, and *Grouping*. For all the experiments, we evaluated the performance of the learners on the test documents described in Table 2.4.

Results

We evaluated the performance of our approach based on three measures: precision, recall, and the F-measure. Given (1) the number of applicable documents N determined by a human expert, (2) the number of correct applicable documents C selected by our approach, and (3) the number of incorrect applicable documents I selected by our approach, we computed the recall ratio as $R = C/N$, the precision ratio as $P = C/(C+I)$, and the F-measure as $F = 2/(1/R + 1/P)$. We reported all these values as percentages in Table 2.5.

Application	Number Applicable Doc.'s	Number Correct	Number Incorrect	Recall %	Precision %	F-Measure %
Car Ads	40	39	2	98	95	96
Obituaries	40	38	2	95	95	95

Table 2.5: Results of the test algorithm in Figure 2.7

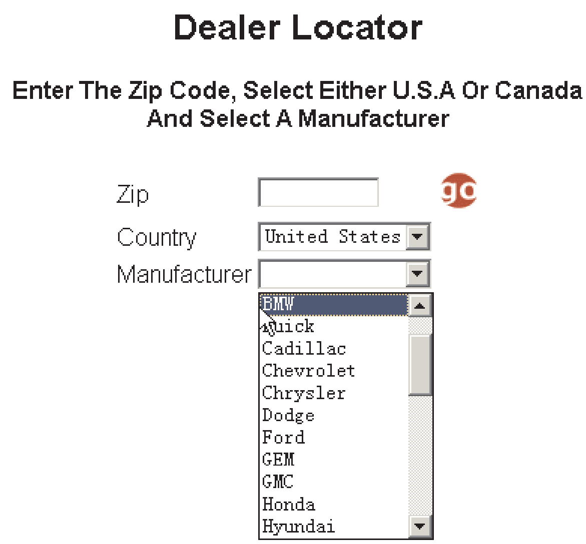
Observe that two negative documents and one applicable document for car ads were classified incorrectly (two incorrect positive responses and one incorrect negative response), and that two applicable documents and two negative documents for obituaries were judged incorrectly (two incorrect negative responses and two incorrect positive responses).

Form Filling and Linked Pages

As explained earlier, in addition to the texts that appear in a test document, we can also exploit auxiliary information such as retrieved documents obtained using form filling to re-check HTML forms. First, we applied a strategy that prefers a better precision ratio. The learner used the algorithm in Figure 2.11 to re-evaluate the application forms that the learner classified as positive responses using the test algorithm in Figure 2.7. Figure 2.16(a) shows a form to locate car dealers, which the algorithm in Figure 2.7 incorrectly classified as a from for car ads. By applying the methods to fill in forms, we retrieved the document in Figure 2.16(b), which contains dealer information rather than car ads. With this additional information, the learner caught the incorrect positive response for the document in Figure 2.16(a) and classified it as negative, nonapplicable to the car-ads application. Thus the learner achieved 98% precision, 98% recall, and an F-measure of 98%. Second, we applied a strategy that prefers a better recall ratio. The learner re-evaluated the negative responses with the available retrieved documents obtained by applying form filling. Figure 2.17 shows both a form in Figure 2.17(a) that the learner classified as nonapplicable using the test algorithm in Figure 2.7 and the applicable document in Figure 2.17(b) by filling in the form in Figure 2.17(a). By using the algorithm in Figure 2.11 to re-evaluate the form in Figure 2.17(a) , the learner caught the incorrect negative response and achieved 95% precision, 100% recall, and an F-measure of 98% for the test set of the car-ads application in Table 2.4.

Figure 2.18 shows an incorrect positive response for the obituaries application, a marriage submission form that contains the keyword "obituary". The form in Figure 2.18, however, requires that the system user manually fill the text fields. Thus the fully automatic classification procedure cannot catch this incorrect positive response.

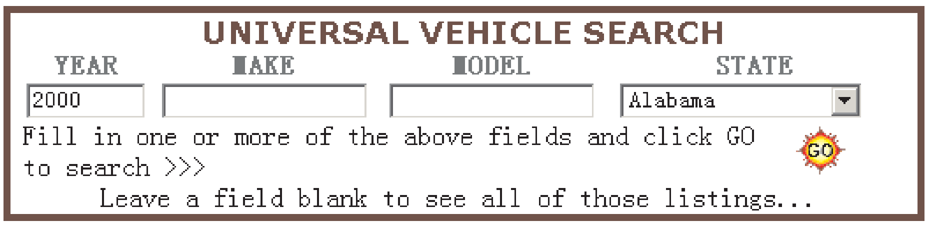
The other auxiliary information we use in our system is information on linked pages. Figure 2.19(a) shows a document that contains only partial obituaries. By applying the algorithm in Figure 2.9, the learner caught the incorrect negative response by considering the linked pages—for example, the applicable document in Figure 2.19(b) for the first link in Figure 2.19(a). Thus the learner achieved 95% precision, 98% recall, and an F-measure of 96%. Note that we included 10 HTML documents containing only partial obituaries similar to the HTML document in Figure 2.19(a) in the test documents. The test result shows that the learner for obituaries missed only one such page without analyzing linked pages. That means that the partial obituaries usually provided enough informative information for the learner to classify them as positive, applicable documents.



(a) Dealer locator

(b) Retrieved document by filling in form in Figure 2.16(a)

Figure 2.16: An incorrect positive response for car ads



(a) Car form

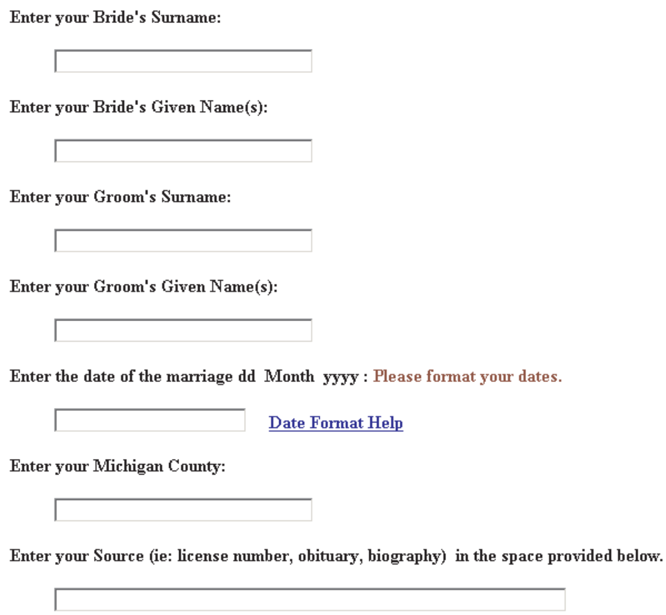
UNIVERSAL SEARCH RESULTS

Select a column to sort these results in alpha-numeric order.
Uncheck the sort box to sort the results in reverse order

YEAR	VEHICLE	LOCATION	MILES	PRICE
2001	Porsche 911 CARRERA 4 Cabriolet - LOADED!	NY Scarsdale	400	$120000
2001	Mitsubishi GALANT GTZ	UT South Jordan	10000	$22000
2001	Ford EXPLORER SPORT	TX Dallas	22000	$18500
2001	Ford ESCAPE XLT	FL Hallandale	2000	$24000
2001	Chrysler PT CRUISER	VA Salem	2500	$20000
2001	Chevrolet IMPALA LS	CA Redding	12000	$20500
2001	Chevrolet CAVALIER LS	VA Rustburg	2000	$10000
2000	Volvo S70	AL Montgomery	36000	$25500
2000	Volkswagen JETTA GLX	VA Falls Church	27000	$21500
2000	Volkswagen CABRIO GLS Convertible	PA Huntingdon Valley	13000	$18500
2000	Toyota TUNDRA	NC Winston Salem	25000	$22500
2000	Toyota TUNDRA	CA Grass Valley	19000	$21000

(b) Retrieved document by filling in form in Figure 2.17(a) (partial)

Figure 2.17: An incorrect negative response for car ads



Enter your Bride's Surname:

Enter your Bride's Given Name(s):

Enter your Groom's Surname:

Enter your Groom's Given Name(s):

Enter the date of the marriage dd Month yyyy : Please format your dates.

Date Format Help

Enter your Michigan County:

Enter your Source (ie: license number, obituary, biography) in the space provided below.

Figure 2.18: An incorrect positive response for obituaries (partial)

Other Incorrect Positives and Negatives

Figure 2.20 shows an incorrect positive response, a motorcycle-sale page, for the car-ads application; and an incorrect positive response, a bibliography of an American hero, for the obituaries application. The motorcycle document contains data for $Year$, $Make$, $Mileage$, $Price$, and $PhoneNr$. The bibliography document contains data such as the person's name, birth and death dates, and relationships including his father and his daughter. Both documents include concepts that largely overlap those specified in the application ontologies. It is difficult for the learners to recognize that documents with significant overlap do not apply to the application ontologies.

The final incorrect response is a negative response for the document in Figure 2.21 for the obituaries application. The learner classified this document as nonapplicable because the density measures obtained based on the entire document text were not high enough to reach the thresholds defined in the single-record tree for obituaries. The reason is that a large amount of irrelevant data appears in the document. In order to catch this incorrect negative response, instead of evaluating the document text that appears in the

06/19: Kanderis, Mike M.

06/16: Torres, Josephine P.
Born: 02/01/1924

06/11: Larsen, Rex Farrell
Born: 04/06/1916

06/11: Oliver, Donna Joy
Born: 09/15/1924

06/10: Anderson, Doug N.
Place of death: Mayfield, Utah
Born: 03/11/1920

06/10: Atkin, Lee Clawson
Place of death: St. George, Utah
Born: 12/11/1932

06/10: Brady, Brittany Kensie
Born: 07/28/1983

06/10: Cass, Betty Louise
Place of death: American Fork, Utah
Born: 10/09/1923

(a) Obituaries that require re-evaluation (partial)

Obituary: Mike M. Kanderis

1925 ~ 2003

Mike M. Kanderis of Littleton, Co, preceded in death by his wife Grace; father of Shane, Dennis and Jennifer; brother of Stella Melonakis and Mary Spencer. Also survived by six grandchildren.

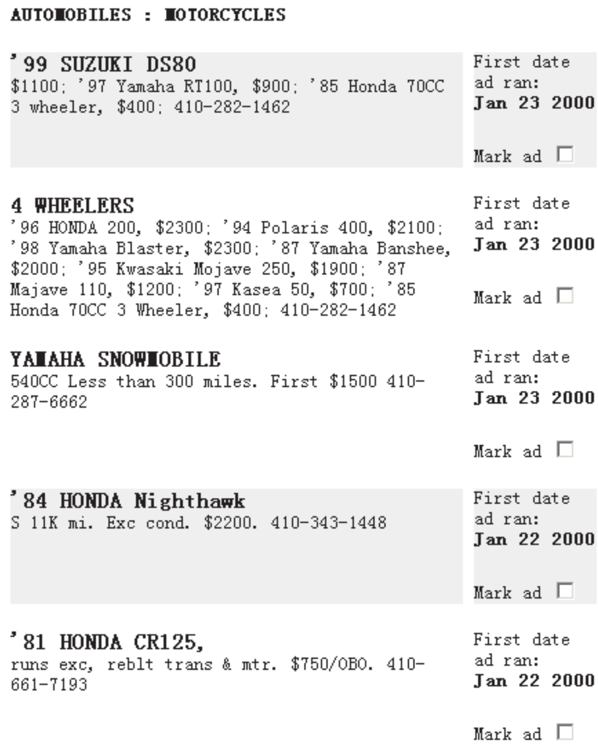
Active member of El Jebel Shrine, ROJ #138 and Sandblasters.

Visitation Wednesday, 4-7 p.m. Horan & McConaty Family Chapel, 31091 So. Wadsworth Blvd. Funeral services Thursday, 2 p.m. St Catherine Greek Orthodox Church, 5555 So. Yosemite, Greenwood Village, CO. Scottish Rite Rose Croix Services Friday 12 noon, at Municipal Cemetery, Grand Junction, CO.

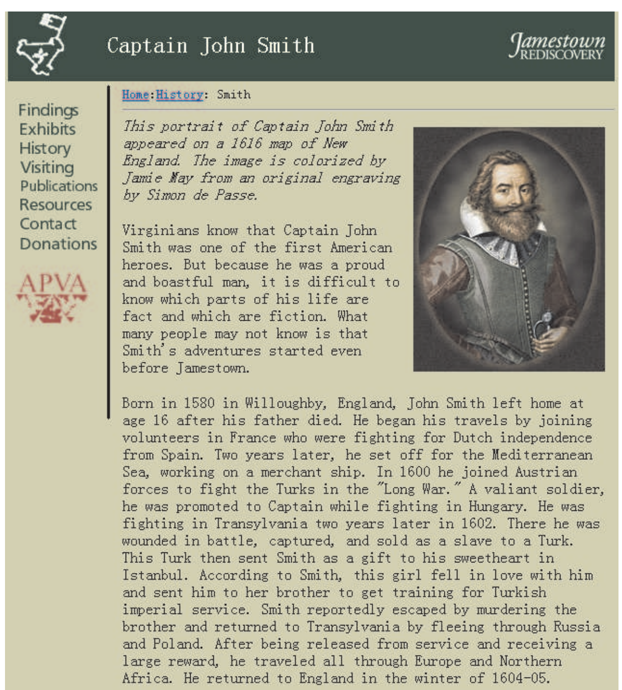
Memorial remembrances to Shrine Hospitals for Children, 4625 W. 50th Ave, Denver, CO 80212

(b) A linked page (partial)

Figure 2.19: An incorrect negative response for obituaries



(a) Motorcycles (partial)



(b) Captain John Smith (partial)

Figure 2.20: Two incorrect responses for car ads and obituaries

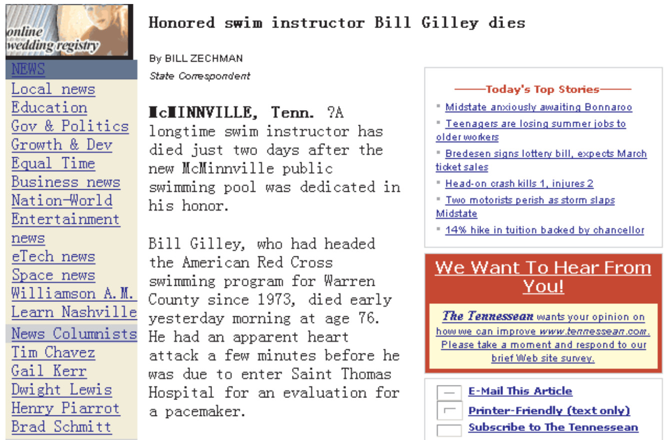


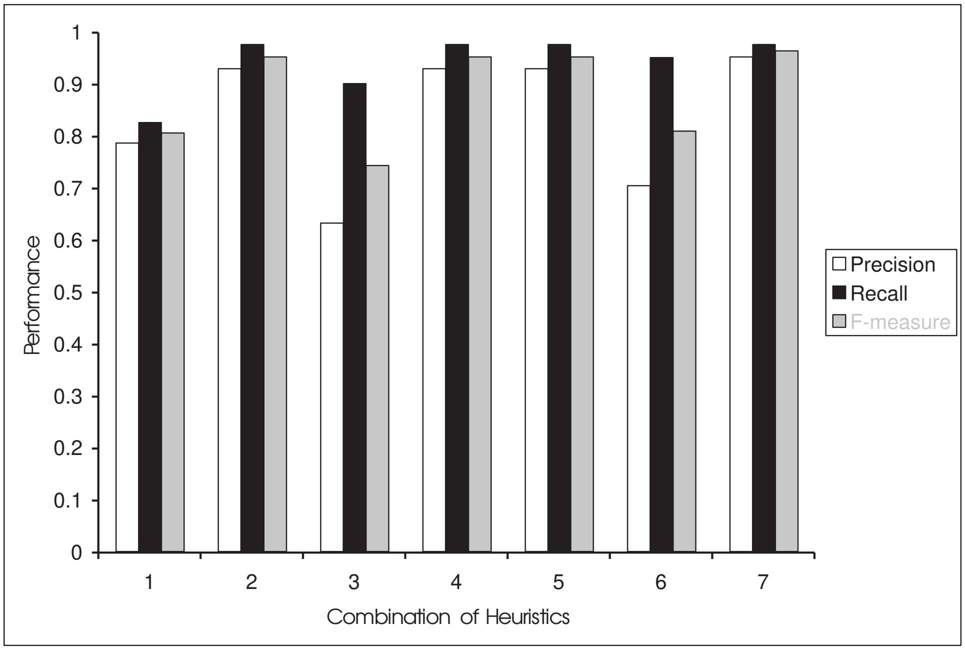
Figure 2.21: Single-record obituary (partial)

entire document, we can evaluate a subpart of the document text that contains the information of interest. For example, we can select the text in Figure 2.21 that describes exactly the singleton obituary for *Bill Gilley*. We manually revised the document and discarded the irrelevant text that appears in the document. Based on the classification model built for obituaries, a positive prediction resulted for the new document. Automatically selecting potentially applicable subcomponents of a document is challenging, and we have not yet resolved the issues and implemented a solution.

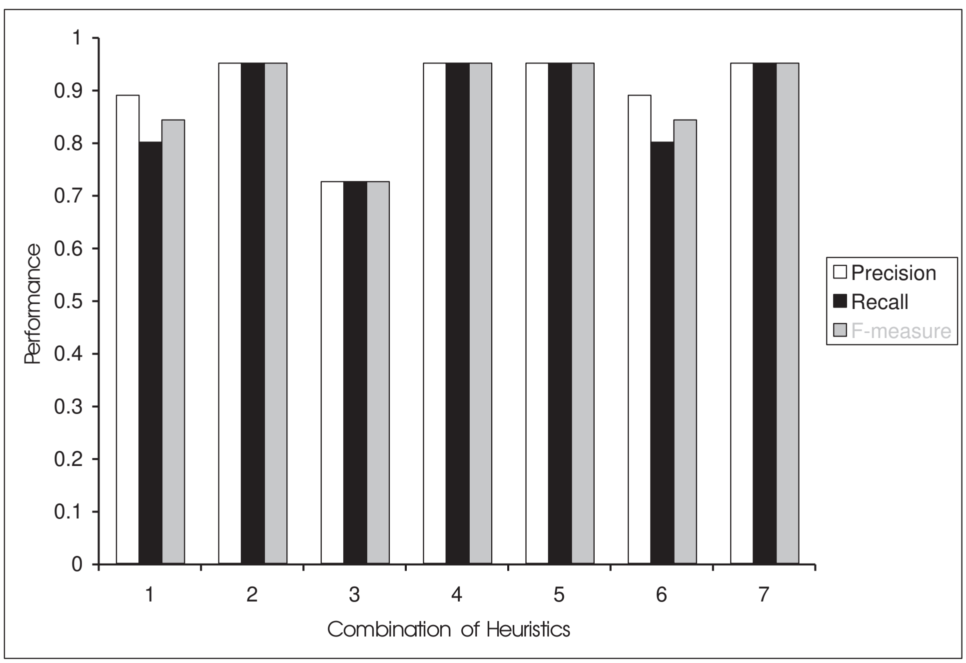
Contributions of Heuristics

We evaluated the performance by applying different combinations of heuristics using the test algorithm in Figure 2.7. Figure 2.22 shows the contribution of the three kinds of heuristics (densities, expected-values, and grouping) to the overall performance in the two applications. The x axis lists the seven combinations of the three kinds of heuristics. Table 2.6 shows the seven combinations, where "+" denotes the heuristic or heuristics in the corresponding column that are in use.

For both applications, Figure 2.22 shows that the set of density measures are important. When the learners exploited only density heuristics in the algorithm of Figure 2.7 to evaluate the test documents for the applications, the learners achieved above 90% for all measures (precision, recall, and F-measure). Using the expected-values heuristic alone,



(a) Cars ads application



(b) Obituaries application

Figure 2.22: Performance comparison of heuristics combinations

Combination	Densities	Expected-Values	Grouping
1			+
2	+		
3		+	
4	+		+
5	+	+	
6		+	+
7	+	+	+

Table 2.6: Combination of three kinds of heuristics

the learners achieved an F-measure of only about 70% for the two applications. Using the grouping heuristic alone, the F-measure obtained by each learner was still less than 85%. Even when the learners used both the expected-values heuristic and the grouping heuristic together, performance for neither application improved. Figure 2.22(a) shows that the learner of the car-ads application achieved the highest F-measure by applying all heuristics together.

We know that the density heuristics are dependent on and sensitive to the specification of the application ontologies. The other two heuristics, expected values and grouping, are also mainly determined by the specification of the application ontology. Thus, when porting to a new application domain, as long as the application ontologies are well defined, our empirical evaluation shows that our approach should be able to recognize applicable HTML documents with high precision, recall, and F-measure. Moreover, as the application ontologies evolve, for example, more lexicons become available for $Make$ or $Color$ is added as a new object set, the performance of the approach is most likely to improve as well.

2.6 Conclusions and Future Work

We presented an approach for recognizing which HTML documents apply to an ontology. Once an application ontology is created, we can use a machine learning algorithm over a set of heuristics to produce a classification model that accurately recognizes which documents apply to the ontology. Results for the tests we conducted showed that the recognition F-measure, precision, and recall were above 95% for both a car-ads application and an obituaries application. We also argued that we can further improve performance by considering linked pages and documents retrieved by submitting default forms.

Our approach is robust, flexible, and scalable. The heuristics, learning algorithms, and training documents used in the approach are extensible. If new heuristics appear useful, we can immediately use them without having to change our fundamental approach. The performance of of a learner can be incrementally improved by adding more training documents. When porting to a new application domain, our approach is able to achieve high precision and recall if the application ontology represents the application domain well. As the application ontology evolves, the learner is likely to improve its performance as well.

In the future, we would like to extend the work proposed in this chapter as follows. First, we would like to test our approach over more applications. Second, we want to extend the heuristics by exploiting more types of knowledge. Third, we plan to make a deeper level analysis for an HTML document such that we can check the applicability based on an appropriate subpart of the document rather than the document text in the entire document. Fourth, we plan to apply a meta-learning strategy to train the learners over training classifiers including C4.5 as described in this dissertation, multivariate analysis [68], and logistic regression[73].

Chapter 3

Automating Schema Mapping for Data Integration

Abstract: Automating schema mapping is challenging. Previous approaches (e.g., [19, 47]) to automating schema mapping focus on computing direct matches between two schemas. Schemas, however, rarely match directly. Thus, to complete the task of schema mapping, we must also compute indirect matches. In this chapter, we present a framework for generating a source-to-target mapping that contains both direct and many indirect matches between a source schema and a target schema. Recognizing expected-data values associated with schema elements and applying schema-structure heuristics are the key ideas needed to compute indirect matches. Experiments we have conducted over several real-world application domains show encouraging results, yielding about 90% precision and recall measures for both direct and indirect matches. Based on our source-to-target mappings, we offer a unified approach for data integration, which combines the advantages of traditional data-integration approaches, mitigates their disadvantages, and provides a viable alternative for flexible and scalable data integration.
Keywords: Schema mapping, data integration, schema integration, data exchange.

3.1 Introduction

Data integration refers to the problem of combining data residing at autonomous and heterogeneous sources and providing users with a unified global schema [11, 34, 71]. Two main concepts constitute the architecture of a data-integration system [71]: wrappers and mediators. A *wrapper* wraps an information source and models the source using a *source schema*. A *mediator* maintains a *global schema* and *mappings* between the source schemas and the global schema. We focus here on data-integration systems that do not materialize data in the global schema.

Currently, there are two main initiatives to integrate data and answer queries without materializing a global schema: Global-as-view (GAV) [17] and Local-as-View (LAV) [32, 42]. In either a GAV or LAV approach, whenever a user poses a query in terms of relations in the global schema, the mediator within the data-integration system uses a *query-reformulation* procedure to translate the query into subqueries that can be executed in sources such that the mediator can collect returned answers from the sources and combine them as the answer to the query.

One of the important applications for data integration is to deal with the explosion of data on the World Wide Web. e-business applications, such as comparison shopping and knowledge-gathering applications, such as vacation planning, raise the following major issues for approaches to data integration.

1. *Heterogeneity*. The sources are autonomous, establishing their own vocabulary and structure.

2. *Scalability*. The number of sources to access and integrate is large.

3. *Continual infusion and change of local information sources*. New sources continually become available and become part of the system. Sources within the system may change frequently.

4. *Query process complexity*. Users frequently pose queries, which can be very complex with respect to the collection of information sources.

5. *Evolution*. As applications evolve, Database Administrators (DBAs) may wish to change the global schema to include some new items of interest.

Most of the existing approaches to data integration [11, 17, 31, 32, 42, 53, 71], however, have addressed only some of these issues. Thus they do not meet the needs of the modern e-business applications.[1] To address these issues, we present an alternative point of view, called a Target-based Integration Query System (TIQS).

TIQS largely depends on schema mapping to resolve the five major issues in a unified framework. Schema mapping is a challenging problem. It takes two schemas as input and produces semantic correspondences between schema elements in the two input schemas [59]. In this dissertation, we assume that we wish to map schema elements from a *source* schema into a *target* schema, where the target schema corresponds to a global schema in data-integration systems. We express each semantic correspondence as a *mapping element*, which binds two schema elements if the two schema elements are semantically equivalent. In its simplest form, a mapping element is a *direct match* that binds a source schema element directly to a target schema element. To date, most research [5, 19, 23, 43, 47, 54, 57] has focused on computing direct matches. Such simplicity, however, is rarely sufficient, and researchers have thus proposed the use of queries over source schemas to bind with target schema elements [6, 52]. In this more complicated form, a mapping element is an *indirect match* that binds a view over the source schema, which is a *virtual* source schema element, to a target schema element through appropriate queries over a source schema.

We assume that all source and target schemas are described using rooted conceptual-model graphs (a conceptual generalization of XML). We augment schemas with a variety of ontological information. For this chapter the augmentations we discuss are WordNet [51], sample data, and regular-expression recognizers. For each application, we construct a lightweight domain ontology [21], which declares typical relationships among objects and the regular-expression recognizers for values and keywords of objects. Based on the

[1] After describing our proposed solution we explain in Section 3.8 by way of comparison how other proposed solutions fail to address one or more of these issues.

graph structure of the relationships and the recognizers, we exploit a broad set of techniques together to settle direct and indirect matches between a source schema and a target schema. As will be seen, regular-expression recognition over sample data and identification of schema structural characteristics are the key ways to detect indirect matches. Based on the discovered matches, the solution proposed here maps a source schema into a target schema through a source-to-target mapping, which is formally specified such that data in the source is ready to load into the target. (Although we may, we need not actually load the data, but can instead pose queries over the virtual target with respect to the data in the sources.)

By applying source-to-target mappings based on a predefined target schema, TIQS provides a unified, scalable approach to data integration, which combines the advantages and avoids the limitations of both GAV and LAV. TIQS has polynomial-time query reformulation, and it is easy to add or modify information sources. DBAs create the target schema and wrap source schemas independently, so that neither the target schema nor the source schemas are contingent respectively on the source schemas or the target schema. When DBAs modify or add new items of interest to the target schema, TIQS applies its schema-mapping techniques to semi-automatically generate or adjust required source-to-target mappings between source schemas and the new target schema. Thus TIQS increases both scalability and usability as compared to previously proposed data-integration approaches.

In this chapter, we offer the following contributions: (1) a way to discover many indirect semantic correspondences between a source schema S and a target schema T as well as the direct correspondences, (2) an extension of relational algebra to express source-to-target mappings, (3) a unified approach to integrate and query data by applying source-to-target mappings, and (3) experimental results of our implementation to show that our solution performs as well as (indeed better than) other approaches for direct matches and also performs exceptionally well for the indirect matches with which we work. We present the details of our contribution as follows. Section 3.2 explains the internal representation of input target and source schemas and needed algebra expressions for schema mappings. Section 3.3 describes a set of basic matching techniques to find potential mapping elements between elements in S and elements in T, and to provide confidence measures between 0 (lowest confidence) and 1 (highest confidence) for each potential match. In Section 3.4, we explain how to combine the confidences output from multiple basic matching techniques by applying a structure-matching technique. Section 3.5 presents a mapping algorithm to settle direct and indirect matches in a source-to-target mapping between S and T. In Section 3.6 we explain how TIQS mitigates the disadvantages and combines the advantages of basic data-integration approaches. Section 3.7 gives experimental results to demonstrate the success of our approach. In Section 3.8 we review related work, and in Section 3.9 we summarize, consider future work, and draw conclusions.

3.2 Internal Representation for Schema Mapping

In a data-integration system, a mediator maintains a target schema, and a wrapper wraps a particular information source as a source schema. It is possible that the wrappers uses their own expression languages to describe the source schema in insolation. In this chapter, we use a conceptual-modeling language OSM-L [20, 44] to describe both source and target schemas, and an extension of the relational algebra to describe views over sources from which we create source-to-target mappings.

3.2.1 Target and Source Schemas

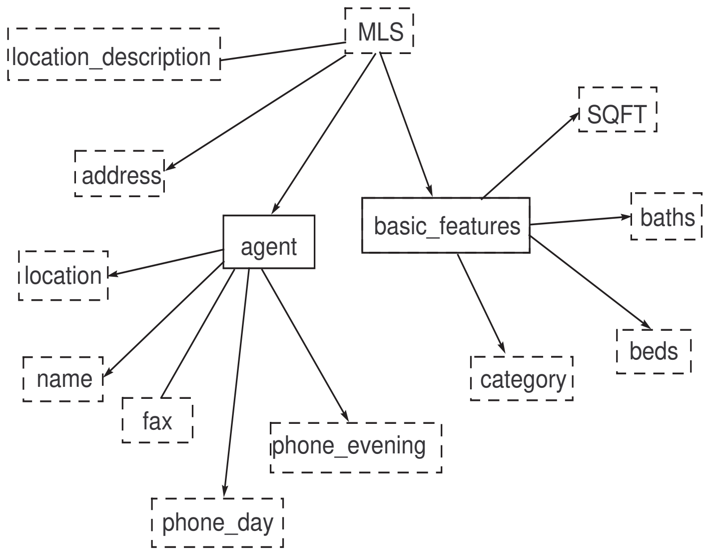
We use rooted graphs to represent both the target schema and the source schemas as conceptual-model specifications. Each conceptual schema has an *object/relationship-model instance* that describes sets of objects, sets of relationships among objects, and constraints over object and relationship sets. In each conceptual schema H, we let O_H denote the set of object sets and R_H denote the set of relationship sets in H. An object set contains either data values or object identifiers, which we respectively call a *lexical object set* or a *nonlexical object set*. A relationship set contains tuples of objects representing relationships connecting object sets. The root node is a designated object of primary interest. Figure 3.1, for example, shows two schema graphs. In a schema graph we denote lexical object sets as dashed boxes, nonlexical object sets as solid boxes, functional relationship sets as lines with an arrow from domain object set to range object set, and nonfunctional relationship sets as lines without arrowheads. For either a target or a source schema, we use an object/relationship-model instance to represent schema-level information in our approach for schema mapping.

An optional component of a conceptual schema is a set of *data frames*, each of which describes the data of a lexical object set. A data frame is like a type which describes data instances, but can be much more expressive. A data-frame description can be as simple as a list of potential values for an object set and can be as complex as a regular-expression specification that represents values for the object set. For target and source schemas in this chapter, data frames are lists of actual or sample values.

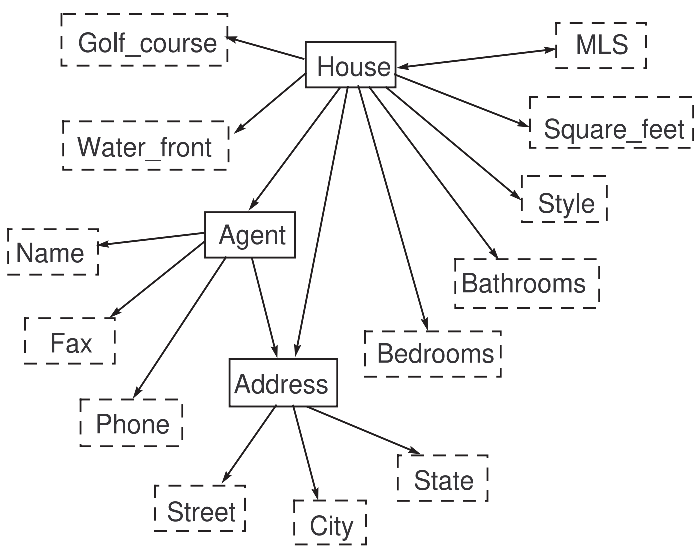
3.2.2 Source-to-Target Mappings

For any schema H, which is either a source schema or a target schema, we let Σ_H denote the union of O_H and R_H in H. Our solution allows a variety of source derived data, including missing generalizations and specializations, merged and split values, transformation of attributes with Boolean indicators into values and vice versa, lexicalization of object identifiers and vice versa, and schema paths as relationships. Therefore, our solution "extends" the source schema elements in Σ_H to include view schema elements, each of which we call a *virtual* object or relationship set. We let V_H denote the extension of Σ_H with derived, virtual object and relationship sets.

We consider a source-to-target mapping between a source schema S and a target schema T as a function f_{ST}. The domain of f_{ST} is V_S, and the range of f_{ST} is Σ_T. Thus we can denote a source-to-target mapping as a function $f_{ST}(V_S) \rightarrow \Sigma_T$. Intuitively, a source-to-target mapping M_i represents inter-schema correspondences between a source schema S_i and a target schema T. If we let Schema 1 in Figure 3.1(a) be the target and let Schema 2 in Figure 3.1(b) be the source, for example, a source-to-target mapping between the two schemas includes a semantic correspondence that declares that the lexical object set $Bedrooms$ in the source semantically corresponds to the lexical object set $beds$ in the target. If we let Schema 1 be the source and Schema 2 be the target, a source-to-target mapping declares that the union of the two sets of values in $phone\_day$ and $phone\_evening$ in the source corresponds to the values for $Phone$ in the target.



(a) Schema 1



(b) Schema 2

Figure 3.1: Conceptual-model graphs for Schema 1 and Schema 2

We represent semantic correspondences between a source schema S and a target schema T as a set of mapping elements. A mapping element is either a *direct match* which binds a schema element in Σ_S ($\subseteq V_S$) to a schema element in Σ_T, or an *indirect match* which binds a virtual schema element in V_S to a target schema element in Σ_T through an appropriate *mapping expression* over Σ_S. A mapping expression specifies how to derive a virtual schema element through manipulation operations over a source schema. We denote a mapping element by $t \sim s \Leftarrow \theta_s(\Sigma_S)$, where $\theta_s(\Sigma_S)$ is a mapping expression that derives a source element s in V_S,[2] and t is a target schema element in Σ_T.

3.2.3 The Algebra for Source-to-Target Mappings

Each object and each relationship set (including derived object and relationship sets) in the target and source schemas are respectively single-attribute or multiple-attribute relations. Thus relational algebra applies directly to the object and relationship sets in a source or target schema. The standard operations, however, are not enough to capture the operations required to express all the needed source-to-target mappings. Thus we extend the relational algebra.

To motivate our use of standard and extended operators, we list the following problems we must face in creating derived object and relationship sets over source schemas.

- *Superset* and *Subset Values*. The object sets, $phone\_day$ and $phone\_evening$ in Schema 1 of Figure 3.1(a) are both subsets of $Phone$ values in Schema 2 of Figure 3.1(b), and the relationship sets $agent$—$phone\_day$ and $agent$—$phone\_evening$ in Schema 1 are both specializations of $Agent$—$Phone$ value pairs in Schema 2. Thus, if Schema 2 is the target, we need the union of the values in $phone\_day$ and $phone\_evening$ and the union of the relationships in $agent$—$phone\_day$ and $agent$—$phone\_evening$ in Schema 1; and if Schema 1 is the target, we should use *Selection* to find a way to separate the day phones from the evening phones and separate the relationships between agents and day phones from those between agents and evening phones.

[2]Note that the mapping expression may be degenerate so that $t \sim s$ is possible.

- *Merged* and *Split Values*. The object sets, $Street$, $City$, and $State$ are separate in Schema 2 and merged as $address$ of $house$ or $location$ of $agent$ in Schema 1. Thus we need to split the values if Schema 2 is the target and merge the values if Schema 1 is the target.

- *Object-Set Name as Value*. In Schema 2 the features $Water\_front$ and $Golf\_course$ are object-set names rather than values. The Boolean values "Yes" and "No" associated with them are not the values but indicate whether the values $Water\_front$ and $Golf\_course$ should be included as description values for $location\_description$ of $house$ in Schema 1. Thus we need to distribute the object-set names as values for $location\_description$ if Schema 1 is the target and make Boolean values for $Water\_front$ and $Golf\_course$ based on the values for $location\_description$ if Schema 2 is the target.

- *Lexicalization and Non-Lexicalization*. In Schema 2, the object identifiers in the object set $House$ represent the house-property objects. One object identifier in $House$ corresponds to one and only one MLS number in the object set MLS. Hence, the nonlexical object set $House$ in Schema 2 potentially matches with the object set MLS in Schema 1. Therefore, we need to lexicalize the object identifiers in $House$ with MLS numbers in Schema 2 if Schema 1 is the target and compute virtual house-property object identifiers based on the MLS numbers in Schema 1 if Schema 2 is the target.

- *Path as Relationship Set*. The path MLS—$basic\_features$—$beds$ in Schema 1 semantically corresponds to the path MLS—$House$—$Bedrooms$ in Schema 2. Thus, if Schema 1 is the target, we need derive two virtual relationship sets corresponding to the target relationship sets MLS—$basic\_features$ and $basic\_features$—$beds$; and if Schema 2 is the target, we need derive two virtual relationship sets corresponding to the target relationship sets $House$—MLS and $House$—$Bedrooms$.

Currently, we use the following operations over source relations to resolve these problems. In the notation, a relation r has a set of attributes, which correspond to the names of lexical or nonlexical object sets; $attr(r)$ denotes the set of attributes in r; and $|r|$ denotes the number of tuples in r. For nonstandard operators, we provide examples to illustrate how to apply the operators over source relations.

- *Standard Operators. Selection* σ, *Union* $\cup$, *Natural Join* $\bowtie$, *Projection* π, and *Rename* ρ.

- *Composition* λ. The λ operator has the form $\lambda_{(A_1,\ldots,A_n),A}r$ where each A_i, $1 \leq i \leq n$, is either an attribute of r or a string, and A is a new attribute. Applying this operation forms a new relation r', where $attr(r') = attr(r) \cup \{A\}$ and $|r'| = |r|$. The value of A for tuple t of row l in r' is the concatenation, in the order specified, of the strings among the A_i's and the string values for attributes among the A_i's for tuple t' of row l in r.

  Let r be the following relation, where $attr(r) = \{House,\ Street,\ City,\ State\}$.

House	Street	City	State
h1	339 Wymt Dr	Provo	Utah
h2	15 S 900 E	Provo	Utah
h3	75 Tiger Ln	Orem	Utah

  Applying the operation $\lambda_{(Street,\text{“, ”},City,\text{“, ”},Street),Address}r$ yields a new relation r', where $attr(r') = \{House,\ Street,\ City,\ State,\ Address\}$.

House	Street	City	State	Address
h1	339 Wymt Dr	Provo	Utah	339 Wymt Dr, Provo, Utah
h2	15 S 900 E	Provo	Utah	15 S 900 E, Provo, Utah
h3	75 Tiger Ln	Orem	Utah	75 Tiger Ln, Orem, Utah

- *Decomposition* γ. The γ operator has the form $\gamma^{R}_{A,A'}r$ where A is an attribute of r, and A' is a new attribute whose values are obtained from A values by applying a routine R. Typically, R extracts a substring from a given string to form part of a decomposition. Repeated application of γ allows us to completely decompose a string. Applying this operation forms a new relation r', where $attr(r') = attr(r) \cup$

$\{A'\}$ and $|r'| = |r|$. The value of A' for tuple t of row l in r' is obtained by applying the routine R to the value of A for tuple t' of row l in r.

Let r be the following relation, where $attr(r) = \{House,\ Address\}$.

House	Address
h1	Provo, Utah
h2	339 Wymt Dr, Provo, Utah
h3	75 Tiger Ln, Orem, Utah

Applying the operation $\gamma^{R}_{Address,Street}r$, where R is a routine that obtains values of $Street$ from values of $Address$, yields a new relation r_1, where $attr(r_1) = \{House,\ Address,\ Street\}$.

House	Address	Street
h1	Provo, Utah	
h2	339 Wymt Dr, Provo, Utah	339 Wymt Dr
h3	75 Tiger Ln, Orem, Utah	75 Tiger Ln

Similarly, applying the operation $\gamma^{R'}_{Address,City}r$, where R' is a routine that obtains values of $City$ from values of $Address$, yields a new relation r_2, where $attr(r_2) = \{House,\ Address,\ City\}$.

House	Address	City
h1	Provo, Utah	Provo
h2	339 Wymt Dr, Provo, Utah	Provo
h3	75 Tiger Ln, Orem, Utah	Orem

- *Boolean* β. The β operator has the form $\beta^{Y,N}_{A,A'}r$, where Y and N are two constants representing Yes and No values in r, A is an attribute of r that has only Y or N values, and A' is a new attribute. The β operator requires the precondition $(attr(r) - \{A\}) \rightarrow \{A\}$. Applying this operation forms a new relation r', where $attr(r') = (attr(r) - \{A\}) \cup \{A'\}$ and $|r'| = |\sigma_{A=Y}r|$. The value of A' for tuple t in r' is the literal string A if and only if there exists a tuple t' in r such that $t'[attr(r) - \{A\}] = t[attr(r) - \{A\}]$ and $t'[A]$ is a Y value.

Let r be the following relation, where $attr(r)$ = $\{House,\ Water\ Front\}$.

House	Water Front
h1	Yes
h2	No
h3	Yes

Applying the operation $\beta^{\text{“}Yes\text{”},\text{“}No\text{”}}_{Water\ Front, Lot\ Description}r$ yields a new relation r', where $attr(r')$ = $\{House, Lot\ Description\}$.

House	Lot Description
h1	Water Front
h3	Water Front

- *DeBoolean* $\reflectbox{$\beta$}$. The $\reflectbox{$\beta$}$ operator has the form $\reflectbox{$\beta$}^{Y,N}_{A,A'}r$, where Y and N are two constants representing Yes and No values, A is an attribute of r, and A' is a new attribute. Applying this operation forms a new relation r', where $attr(r') = (attr(r) - \{A\}) \cup \{A'\}$ and $|r'| = |\pi_{attr(r)-\{A\}}r|$. The value of A' for tuple t in r' is Y if and only if there exists a tuple t' in r such that $t'[attr(r) - \{A\}] = t[attr(r) - \{A\}]$ and $t'[A]$ is the literal string A', or is N if and only if there does not exist a tuple t' in r such that $t'[attr(r) - \{A\}] = t[attr(r) - \{A\}]$ and $t'[A]$ is the literal string A'.

  Let r be the following relation, where $attr(r)$ = $\{House,\ Lot\ Description\}$.

House	Lot Description
h1	Water Front
h1	Golf Course
h1	Mountain View
h2	Water Front
h3	Golf Course

Applying the operation $\beta^{\text{``}Yes\text{''},\text{``}No\text{''}}_{Lot\ Description, Water\ Front} r$ yields a new relation r', where $attr(r')$ = $\{House, Water\ Front\}$.

House	Water Front
h1	Yes
h2	Yes
h3	No

Similarly, applying the operation $\beta^{\text{``}x\text{''},\text{``''}}_{Lot\ Description, Golf\ Course} r$ yields a new relation r'', where $attr(r'')$ = $\{House, Golf\ Course\}$.

House	Golf Course
h1	x
h2	
h3	x

- *Skolemization* φ. The φ operator has the form $\varphi_{f_A}(r)$, where f_A is a skolem function, and A is a new attribute. Applying this operation forms a new relation r', where $attr(r') = attr(r) \cup \{A\}$ and $|r'| = |r|$. The value of A for tuple t of row l in r' is a functional term that computes a value by applying the skolem function f_A over tuple t' of row l in r.

  Let r be the following relation, where $attr(r)$ = $\{House\}$.

House
h1
h2
h3

  Applying the operation $\varphi_{f_{Basic\ Features}} r$ yields a new relation r', where $attr(r')$ = $\{House, Basic\ Features\}$.

House	Basic Features
h1	$f_{Basic\ Features}(h1)$
h2	$f_{Basic\ Features}(h2)$
h3	$f_{Basic\ Features}(h3)$

3.3 Matching Techniques

In this section we explain our three basic techniques for schema matching: (1) terminological relationships (e.g., synonyms and hypernyms), (2) data-value characteristics (e.g., string lengths and alphanumeric ratios), (3) domain-specific, regular-expression matches (i.e. the appearance of expected strings), and (4) structure (e.g., structural similarities). For the first two techniques we obtain vectors of measures for the features of interest and then apply machine learning over these feature vectors to generate a decision rule and a measure of confidence for each generated decision. We use C4.5 [58] as our decision-rule and confidence-measure generator. For the third technique, we analyze data values based on a domain ontology to compute confidences and discover indirect matches as well as direct matches.

3.3.1 Terminological Relationships

To match element names, we use WordNet [28, 51], which organizes English words into synonym and hypernym sets. Other researchers have also suggested using WordNet to match attributes (e.g., [5, 13]), but have given few, if any, details.

Initially we investigated the possibility of using 27 available features of WordNet in an attempt to match a token A appearing in the name of a source schema element s with a token B appearing in the name of an target schema element t. The C4.5-generated decision tree, however, was not intuitive.[3] We therefore introduced some bias by selecting only those features we believed would contribute to a human's decision to declare a potential attribute match, namely (f0) same word (1 if $A = B$ and 0 otherwise), (f1) synonym (1 if "yes" and 0 if "no"), (f2) sum of the distances of A and B to a common hypernym ("is kind of") root (if A and B have no common hypernym root, the distance is defined as a maximum number in the algorithm), (f3) the number of different common hypernym roots of A and B, and (f4) the sum of the number of word senses of A and B. For our training data we used 222 positive and 227 negative A-B pairs selected from attribute names found in database schemas, which were readily available to us, along with synonym names found

[3] An advantage of decision-tree learners over other machine learning (such as neural nets) is that they generate results whose reasonableness can be validated by a human.

```
f3 <= 0: NO (222.0/26.0)
f3 > 0
|  f2 <= 2: YES (181.0/3.0)
|  f2 > 2
|  |  f4 <= 11
|  |  |  f2 <= 5: YES (15.0/5.0)
|  |  |  f2 > 5: NO (14.0/6.0)
|  |  f4 > 11: NO (17.0/2.0)
```

Figure 3.2: Generated WordNet rule by applying the C4.5 algorithm

in dictionaries. Figure 3.2 shows the resulting decision tree. Surprisingly, neither f0 (same word) nor f1 (synonym) became part of the decision rule. Feature f3 dominates—when WordNet cannot find a common hypernym root, the words are not related. After f3, f2 makes the most difference—if two words are closely related to the same hypernym root, they are a good potential match. (Note that f2 covers f0 and f1 because both identical words and direct synonyms have zero distance to a common root; this helps mitigate the surprise about f0 and f1.) Lastly, if the number of senses is too high (f4 $>$ 11), a pair of words tends to match almost randomly; thus the C4.5-generated rule rejects these pairs and accepts fewer senses only if pairs are reasonably close (f2 <= 5) to a common root.

The parenthetical numbers (x/y) following "YES" and "NO" for a decision-tree leaf L give the total number of training instances x classified for L and the number of incorrect training instances y classified for L. Based on the trained decision rule in Figure 3.2, we compute a confidence value, denoted $conf_1(s,t)$, where s is a source schema element and t is a target schema element. However, we want the feature f0 (same word) to dominate the others and assign a perfect confidence value (1.0) for two tokens if f0 holds. When schema element names are abbreviations, we expand them so that WordNet can recognize them. If the names of both s and t are single-word tokens, the computation $conf_1(s,t)$ is straightforward based on the decision rule when f0 does not hold. For a "YES" leaf L, we compute confidence factors by the formula (x-y)/x where x is the total number of training instances classified for L and y is the number of incorrect training instances classified for

	MLS	bath.	bed.	cat.	SQ.	loc._desc.	basic._feat.	agent	fax	ph._day	ph._even.	name	loc.	addr.
House	0.11	0.12	0.12	0.11	0.11	0.12	0.11	0.11	0.11	0.11	0.11	0.98	0.12	0.11
Bathrooms	0.11	0.98	0.98	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11
Bedrooms	0.11	0.98	0.98	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11
MLS	0.98	0.11	0.11	0.11	0.11	0.43	0.11	0.11	0.11	0.11	0.11	0.43	0.11	0.43
Square_feet	0.11	0.11	0.11	0.11	0.98	0.11	0.11	0.11	0.43	0.27	0.27	0.12	0.11	0.11
Water_front	0.11	0.11	0.11	0.12	0.12	0.12	0.12	0.11	0.11	0.11	0.11	0.12	0.11	0.12
Golf_course	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11
Address	0.43	0.11	0.11	0.11	0.11	0.98	0.12	0.11	0.11	0.11	0.11	0.12	0.11	0.98
Agent	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.98	0.11	0.11	0.11	0.11	0.11	0.11
Fax	0.11	0.11	0.11	0.11	0.43	0.11	0.11	0.11	0.98	0.67	0.67	0.11	0.11	0.11
Phone	0.11	0.11	0.11	0.11	0.43	0.11	0.11	0.11	0.67	0.98	0.98	0.98	0.11	0.11
Name	0.43	0.11	0.11	0.11	0.12	0.43	0.11	0.11	0.11	0.98	0.98	0.98	0.11	0.12
Street	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.98	0.11	0.43	0.11	0.11
State	0.11	0.11	0.11	0.11	0.12	0.11	0.11	0.11	0.11	0.11	0.11	0.12	0.11	0.11
City	0.11	0.11	0.11	0.11	0.67	0.11	0.11	0.11	0.11	0.11	0.11	0.43	0.11	0.11
Style	0.11	0.11	0.11	0.98	0.43	0.98	0.12	0.11	0.43	0.43	0.43	0.11	0.11	0.98

Figure 3.3: WordNet confidence-value matrix

L. For a "NO" leaf, the confidence factor is 1-(x-y)/x, which converts "NO's" into "YES's" with inverted confidence values. If a schema element name is a phrase instead of a single-word token, we select nouns from the phrase. Then if either s or t has a name consisting of multiple nouns, we use an injective, greedy match algorithm to locate the potential matching nouns between the name phrases of s and t. The algorithm takes the best matching pair of words and then eliminates the matched words from the name phrases in s and t before selecting the next best pair, and so forth. We compute $conf_1(s, t)$ as the average of the confidence values collected from the potential matching tokens obtained from the injective, greedy algorithm.

Assume that Schema 1 in Figure 3.1(a) is a source schema, and Schema 2 in Figure 3.1(b) is a target schema. Figure 3.3 shows a confidence-value matrix generated by the decision rule in Figure 3.2 for the target and source schemas. The schema elements along the top are source schema elements taken from Schema 1.[4] The schema elements on the left are target schema elements taken from Schema 2. Observe, for example, that the confidence values $conf_1(agent, Agent)$, $conf_1(beds, Bedrooms)$, $conf_1(baths, Bathrooms)$, $conf_1(phone\_day, Phone)$, and $conf_1(phone\_evening, Phone)$ are high as they should be. Observe, however, that the two confidence values $conf_1(location\_description, Golf\_course)$ and $conf_1(location\_description, Water\_front)$ are low, even though "Golf_course" and "Water_front" around a house property are two kinds of "location_description";

[4] In order to fit in the page, we use abbreviations for schema-element names in the source.

and the confidences $conf_1(category, Style)$, $conf_1(location\_description, Style)$, $conf_1(address, Style)$ are high based on the WordNet hierarchical structure, even though the object sets do not semantically correspond with each other. As we shall see, however, other techniques can sort out and eliminate these anomalies.

3.3.2 Data-Value Characteristics

Previous work in [43] shows that characteristics among data values can successfully help match elements by considering such characteristics as string-lengths and alphabetic/non-alphabetic ratios of alphanumeric data and means and variances of numerical data. We use features similar to those in [43], but generate a C4.5 decision rule rather than a neural-net decision rule. Based on the decision rule, which turns out to be lengthy but has a form similar to the decision tree in Figure 3.2, we generate a confidence value, denoted $conf_2(s, t)$, for each element pair (s, t) of schema elements that have data values available.

Figure 3.4 shows a confidence-value matrix generated by the decision rule using data values associated with Schema 1 in Figure 3.1(a) as a source schema and Schema 2 in Figure 3.1(b) as a target schema. Note that in Figure 3.4 there are several nonlexical object sets whose values are object identifiers in Schema 1 and Schema 2. An *NA* in the matrix denotes that the object identifiers associated with either the source object set in a column or the target object set in a row are not applicable for value analysis. Observe that the confidence values such as $conf_2(beds, Bedrooms)$, $conf_2(baths, Bathrooms)$, $conf_2(phone\_day, Phone)$, and $conf_2(fax, Fax)$ are high, as expected. Observe, however, several high confidence values produced are not correct. For example, Fax in the target and $phone\_day$ in the source tend to look alike according to the value characteristics measured, an incorrect match which needs other techniques to find the difference. Interestingly, the house location description in $location\_description$, the category in $category$, and the house address in $address$ of the source schema do not have high similar value characteristics with style values in $Style$ of the target schema. This is because either their string length ratios or their alpha/non-alpha ratios are vastly different, as they should be.

	MLS	bath.	bed.	cat.	SQ.	loc._desc.	basic._feat.	agent	fax	ph._day	ph._even.	name	loc.	addr.
House	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Bathrooms	0.33	0.86	0.33	0.08	0.86	0.08	NA	NA	0.33	0.33	0.33	0.08	0.33	0.33
Bedrooms	0.33	0.31	0.86	0.08	0.33	0.08	NA	NA	0.33	0.31	0.31	0.08	0.33	0.33
MLS	0.86	0.67	0.31	0.08	0.33	0.08	NA	NA	0.33	0.67	0.67	0.08	0.33	0.33
Square_feet	0.33	0.86	0.33	0.08	0.86	0.08	NA	NA	0.91	0.33	0.33	0.08	0.33	0.33
Water_front	0.08	0.05	0.05	0.33	0.05	0.33	NA	NA	0.05	0.08	0.08	0.33	0.08	0.08
Golf_course	0.08	0.05	0.05	0.33	0.05	0.33	NA	NA	0.05	0.08	0.08	0.33	0.08	0.08
Address	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Agent	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Fax	0.91	0.33	0.33	0.08	0.91	0.08	NA	NA	0.86	0.86	0.86	0.08	0.33	0.33
Phone	0.31	0.33	0.33	0.08	0.91	0.08	NA	NA	0.86	0.86	0.86	0.08	0.33	0.33
Name	0.08	0.08	0.08	0.86	0.08	0.86	NA	NA	0.08	0.1	0.1	0.86	0.1	0.1
Street	0.91	0.33	0.33	0.91	0.33	0.1	NA	NA	0.86	0.86	0.86	0.91	0.31	0.91
State	0.08	0.05	0.05	0.33	0.05	0.33	NA	NA	0.05	0.08	0.08	0.33	0.08	0.08
City	0.08	0.08	0.08	0.91	0.08	0.33	NA	NA	0.08	0.1	0.1	0.33	0.08	0.08
Style	0.91	0.08	0.08	0.67	0.08	0.67	NA	NA	0.08	0.08	0.08	0.86	0.08	0.08

Figure 3.4: Value-characteristics confidence-value matrix

3.3.3 Expected Data Values

Whether expected values appear in a set of data provides yet another a clue to which elements match. For a specific application, we can specify a lightweight domain ontology [21], which includes a set of concepts and relationships among the concepts, and associates with each concept a set of regular expressions that matches values and keywords expected to appear for the concept. Then using techniques described in [21], we can extract values from sets of data associated with source and target elements and categorize their data-value patterns based on the regular expressions declared for application concepts. The derived data-value patterns and the declared relationship sets among concepts in the domain ontology can help discover both direct and indirect matches for schema elements. Figure 3.5 shows the regular expressions using the Perl syntax we specified for two concepts in a lightweight domain ontology for real-estate applications.[5]

We declare the concepts and relationship sets in our lightweight domain ontologies independently of any target and source schemas. We call them lightweight for two reasons. (1) The construction of concepts and relationships is not the same as the construction of a conceptual schema in GAV data-integration approaches [71] for integrating heterogeneous information sources. A GAV data-integration system maintains a global schema, and the system needs to update the global schema when new information sources become available. Thus the GAV approach requires that the global schema be complete

[5] [37] provided a tool to create the regular-expression specifications.

```
View matches [15] case insensitive
    constant
        { extract "\bmountain\sview\b"; },
        { extract "\bocean\sview\b"; },
        { extract "\briver\sview\b"; },
        { extract "\bbay\sview\b"; },
        { extract "\bharbor\sview\b"; },
        { extract "\bwater\sview\b"; },
        { extract "\bpanoramic\sview\b"; },
        { extract "\bcity\slight(s)?\b"; },
        { extract "\bcity\sview\b"; },
        { extract "\bvalley\sview\b"; },
        { extract "\bgardon\sview\b"; },
        { extract "\bpool\sview\b"; },
        { extract "\bgolf\s*course\sview\b"; },
        { extract "\bcoastline\sview\b"; },
            ...
        { extract "\bgreenbelt\sview\b"; };
    keyword
        "\bview(s)?\b";
End;
Phone matches [15] case insensitive
    constant
        { extract "\b\d{3}-\d{4}\b"; }, – nnn-nnnn
        { extract "\b\(\d{3}\)\s*\d{3}-\d{4}\b"; }, – (nnn) nnn-nnnn
        { extract "\b\d{3}-\d{3}-\d{4}\b"; }, – nnn-nnn-nnnn
        { extract "\b\d{3}\\\d{3}-\d{4}\b"; }, –nnn\nnn-nnnn
        { extract "\b1-\d{3}-\d{3}-\d{4}\b"; }; – 1-nnn-nnn-nnnn
    Keyword
        "\bcall\b;
End;
```

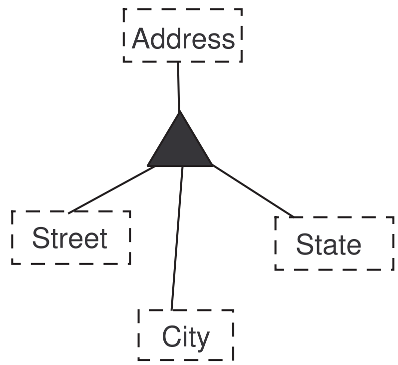
Figure 3.5: Example of regular expressions specified for concepts in lightweight domain ontologies

in the sense that it embodies all the contents in the underlying information sources. We neither require nor expect that the knowledge declared in an application domain ontology to be complete for the application. Moreover, (2) the objective of the regular expressions declaring expected values for application concepts is to discover corresponding concepts, not to extract items of interest [21]. Since the domain ontology need not be as complete or as exact as the declarations for a data-extraction ontology, we see our domain ontologies as being lightweight.

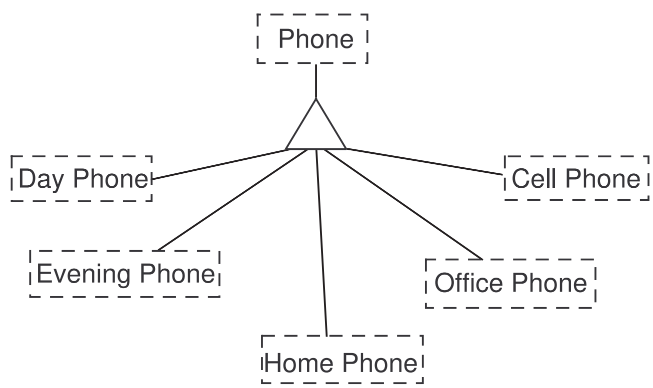
Figure 3.6 shows three components in our real-estate domain ontology, which we used to automate matching of the two schemas in Figure 3.1 and also for matching real-world schemas in the real-estate domain in general. The three components include an address component specifying $Address$ as potentially consisting of $State$, $City$, and $Street$;[6] a phone component specifying $Phone$ as a possible superset of $Day\ Phone$, $Evening\ Phone$, $Home\ Phone$, $Office\ Phone$, and $Cell\ Phone$;[7] and a lot-feature component specifying $Lot\ Feature$ as a possible superset of $View$ and $Lot\ Size$ values and individual values $Water\ Front$, $Golf\ Course$, etc.[8] Behind a dashed box (or individual value), a regular-expression recognizer [21] describes the expected-data values for a potential application concept. The ontology explicitly declares that (1) the expected values in $Address$ match with a concatenation of the expected values for $Street$, $City$ and $State$; (2) the set of values associated with $Phone$ is a superset of the values in $Day\ Phone$, $Evening\ Phone$, $Home\ Phone$, $Office\ Phone$, and $Cell\ Phone$; and (3) the set of values associated with $Lot\ Feature$ is a superset of the values associated with the set of $View$ values, the set of $Lot\ Size$ values, the singleton-sets including $Water\ Front$, $Golf\ Course$, $Wooded$, $Fenced\,Yard$, $Cul-de-sac$, etc.

Provided with the domain ontology just described and a set of data values for elements in Schema 1 in Figure 3.1(a) and Schema 2 in Figure 3.1(b), we can discover indirect matches as follows. (We first introduce the idea with examples and then more formally explain how this works in general.)

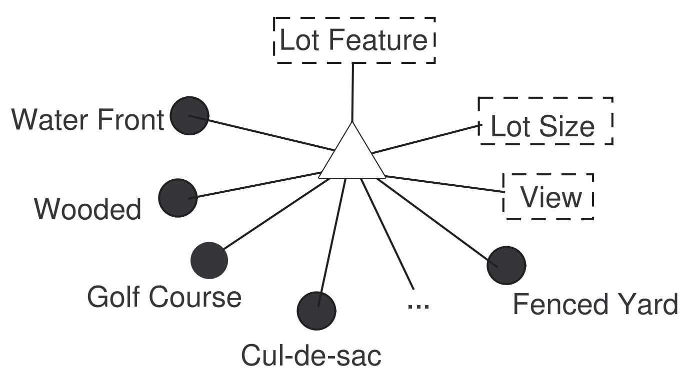
[6]Filled-in (black) triangles denote aggregation ("part-of" relationships).
[7]Open (white) triangles denote generalization/specialization ("ISA" supersets and subsets).
[8]Large black dots denote individual objects or values.



(a) Address



(b) Phone



(c) Lot Feature

Figure 3.6: Application domain ontology (partial)

1. ***Merged*** and ***Split Values***. Based on the $Address$ declared in the ontology in Figure 3.6, the recognition-of-expected-values technique [21] can help detect that (1) the values of $address$ in Schema 1 of Figure 3.1(a) match with the ontology concept $Address$, and (2) the values of $Street$, $City$, and $State$ in Schema 2 of Figure 3.1(b) match with the ontology concepts $Street$, $City$, and $State$ respectively. Thus, if Schema 1 is the source and Schema 2 is the target, we can use *Decomposition* over $address$ in the source to derive three virtual object sets such that the three virtual object sets match with $Street$, $City$, and $State$ respectively in the target. If we let Schema 2 be the source and Schema 1 be the target, based on the same information, we can identify an indirect match that declares a virtual object set derived by applying the ***Composition*** operation over the source to merge values in $Street$, $City$, and $State$ to directly match with $address$ in the target.[9]

2. ***Superset*** and ***Subset Values***. Based on the specification of the regular expression for $Phone$, the schema elements $phone\_day$ and $phone\_evening$ in Schema 1 of Figure 3.1(a) match with the concepts $Day\ Phone$ and $Evening\ Phone$ respectively, and $Phone$ in Schema 2 of Figure 3.1(b) also matches with the concept $Phone$. $Phone$ in the ontology explicitly declares that its set of expected values is a superset of the expected values of $Day\ Phone$ and $Evening\ Phone$. Thus we are able to identify the indirect matching schema elements between $Phone$ in Schema 2 and $phone\_day$ and $phone\_evening$ in Schema 1. If Schema 1 is the source and Schema 2 is the target, we can apply a ***Union*** operation over Schema 1 to derive a virtual schema element $Phone'$, which can directly match with $Phone$ in Schema 2. If Schema 2 is the source and Schema 1 is the target, we may be able to recognize keywords such as *day-time*, *day*, *work phone*, *evening*, and *home* associated with each listed phone in the source. If so, we can use a *Selection* operator to sort out which phones belong in which specialization (if not, a human expert may not be able to sort these out either).

[9] When applying the manipulation operations over sources in data-integration applications, the data-integration system requires routines to merged/split values so that correctly retrieving data from sources.

3. *Object-Set Name as Value*. Because regular-expression recognizers can recognize schema element names as well as values, the recognizer for $Lot\ Feature$ recognizes names such as $Water\_front$ and $Golf\_course$ in Schema 2 of Figure 3.1(b) as values. Moreover, the recognizer for $Lot\ Feature$ can also recognize data values associated with $location\_description$ in Schema 1 of Figure 3.1(a) such as "***Mountain View***", "***City Overlook***", and "***Water-Front Property***". Thus, when Schema 1 is the source and Schema 2 is the target, whenever we match a target-schema-element name with a source $location\_description$ value, we can declare "Yes" as the value for the matching target concept. If, on the other hand, Schema 2 is the source and Schema 1 is the target, we can declare that the schema element name should be a value for $location\_description$ for each "Yes" associated with the matching source element.

We now more formally describe these three types of indirect matches. Let c_i be an application concept, such as $Street$, and consider a concatenation of concepts such as $Address$ components. Suppose the regular expression for concept c_i matches the first part of a value v for a schema element and the regular expression for concept c_j matches the last part of v, then we say that the concatenation $c_i \circ c_j$ matches v. In general, we may have a set of concatenated concepts C_s match a source element s and a set of concatenated concepts C_t match a target element t. For each concept in C_s or in C_t, we have an associated hit ratio. Hit ratios give the percentage of s or t values that match (or are included in at least some match) with the values of the concepts in C_s or C_t respectively. We also have a hit ratio r_s associated with C_s that gives the percentage of s values that match the concatenation of concepts in C_s, and a hit ratio r_t associated with C_t that gives the percentage of t values that match the concatenation of concepts in C_t. To obtain hit ratios for Boolean fields recognized as schema-element names, we distribute the schema-element names over all the Boolean fields that have "Yes" values.

We decide if s matches with t directly or indirectly by comparing C_s and C_t if the hit ratios r_s and r_t are above an accepted threshold. If C_s equals C_t, we declare a *direct* match (s, t). Otherwise, if $C_s \supset C_t$ ($C_s \subset C_t$), we derive an *indirect* match (s, t) through a *Decomposition* (*Composition*) operation. If both C_s and C_t contain one individual concept c_s and c_t respectively, and if the values of concept c_s (c_t) are declared as a subset of the values of concept c_t (c_s), we derive an *indirect* match (s, t) through a *Union* (*Selection*) operation. When we have schema-element names as values, distribution of the name over the Boolean value fields converts these schema elements into standard schema elements with conventional value-populated fields. Thus no additional comparisons are needed to detect direct and indirect matches when schema-element names are values. We must, however, remember the Boolean conversion for both source and target schemas to correctly derive indirect matches.

We compute the confidence value for a mapping (s, t), which we denoted as $conf_3(s,t)$, as follows. If we can declare a direct match or derive an indirect match through manipulating *Union*, *Selection*, *Composition*, *Decomposition*, *Boolean*, and *DeBoolean* operators for (s, t), we output the highest confidence value 1.0 for $conf_3(s,t)$. Otherwise, we construct two vectors v_s and v_t whose coefficients are hit ratios associated with concepts in C_s and C_t. To take the partial similarity between v_s and v_t into account, we calculate a VSM [2] cosine measure $cos(v_s, v_t)$ between v_s and v_t, and let $conf_3(s,t)$ be ($cos(v_s, v_t)$ $\times$ $(r_s + r_t)/2$).

Figure 3.7 shows the matrix containing confidence values computed based on expected-data values using Schema 1 in Figure 3.1(a) as a source schema and Schema 2 in Figure 3.1(b) as a target schema. Observe that the technique correctly identifies the indirect matches between $location\_description$ in the source and $Golf\_course$ and $Water\_front$ in the target, between $phone\_day$ and $phone\_evening$ in the source and $Phone$ in the target, and between $address$ and $location$ in the source and $Street$, $City$, and $State$ in the target. Once again note that the object identifiers associated with nonlexical object sets in both target and source schemas are inapplicable for the expected-data-values analysis. Furthermore, for this example, we did not include the specifications for expected-data values of “bedrooms” and “bathrooms” in our lightweight ontology. The values for $Bedrooms$ and $Bathrooms$ in the target and the values for $beds$ and $baths$ in the source do not match any concept in the domain ontology. If one set of data values corresponds to the expected-data values specified for a concept and another set of data values does not

	MLS	bath.	bed.	cat.	SQ.	loc._desc.	basic_feat.	agent	fax	ph._day	ph._even.	name	loc.	addr.
House	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Bathrooms	0.0	NA	NA	NA	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Bedrooms	0.0	NA	NA	NA	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
MLS	1.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Square_feet	0.0	0.0	0.0	0.0	1.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Water_front	0.0	0.0	0.0	0.0	0.0	1.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Golf_course	0.0	0.0	0.0	0.0	0.0	1.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Address	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Agent	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Fax	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	1.0	0.0	0.0	0.0	0.0	0.0
Phone	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	1.0	1.0	0.0	0.0	0.0
Name	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	1.0	0.0	0.0
Street	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	1.0	1.0
State	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	1.0	1.0
City	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	1.0	1.0
Style	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0

Figure 3.7: Expected-data-values confidence-value matrix

correspond to any concept in the ontology, the confidence is 0.0. For example, the confidence $conf_3(baths, Phone)$ is 0.0 because the values for $Phone$ in the target correspond to the concept *Phone* in the ontology, but the values for $baths$ in the source do not. If neither values of a pair corresponds to any concept specification in the ontology,[10] the entry is *NA*. For example, the *NA* for the pair ($baths$, $Bathrooms$) denotes that the data values for neither $baths$ in the source nor $Bathrooms$ in the target match any concept in the lightweight real-estate domain ontology. If the domain ontology is not complete with respect to an application, our approach needs other matching techniques to discover matches that are not discovered through comparing expected-data values.

3.4 Structure Matching

The three matching techniques including the terminological relationships in Section 3.3.1, the value characteristics in Section 3.3.2, and the expected-data values in Section 3.3.3 compare only object sets between a source schema and a target schema. In addition to object-set matches, structure matching applies to schema structural properties to resolve relationship-set matches between two schemas. The object-set matches themselves are not enough to provide access paths for retrieving data from the source. For example,

[10] We are not able to compare the expected-data values without the help of the domain ontology.

assume that we let Schema 1 of Figure 3.1(a) be a target schema and Schema 2 of Figure 3.1(b) be a source schema. Based on the terminological relationships, value characteristics, and expected-data values, we obtain object-set matches such as the match between MLS and MLS and the match between $Bedrooms$ and $beds$. Without relationship-set matches, however, it may be impossible to correctly answer a user query such as "Find houses with 4 bedrooms" because the query requires the system to match the semantically equivalent relationships in the source with the relationships between MLS and $beds$ in the target.

In a source-to-target mapping, a mapping element $t \sim s \Leftarrow \theta_s(\Sigma_S)$ is either an object-set match or a relationship-set match. Since the mapping element declares that a source schema element s, which is either an element in the source or a view over the source, is semantically equivalent to a target element t, we can access the data of s through the target element t. Intuitively, a source-to-target mapping between a source schema and a target schema describes all the needed access paths to retrieve data facts from the source for matching target elements.

When comparing structural properties of a target schema and a source schema, we apply a top-down strategy. At the top level, we compute semantic correspondences between abstract components of the target and source schemas. Each of the components for all schemas is composed of a set of object sets and relationship sets among the object sets. The structure-matching technique determines the composition of abstract components for target and source schemas based on schema structural constraints and available confidence values for potential object-set matches from the terminological relationship sets, the value characteristics, and the expected-data values. Then, at the bottom level, with the guide of compatible components between the target and source schemas, we compute the finer-level correspondences between object and relationship sets.

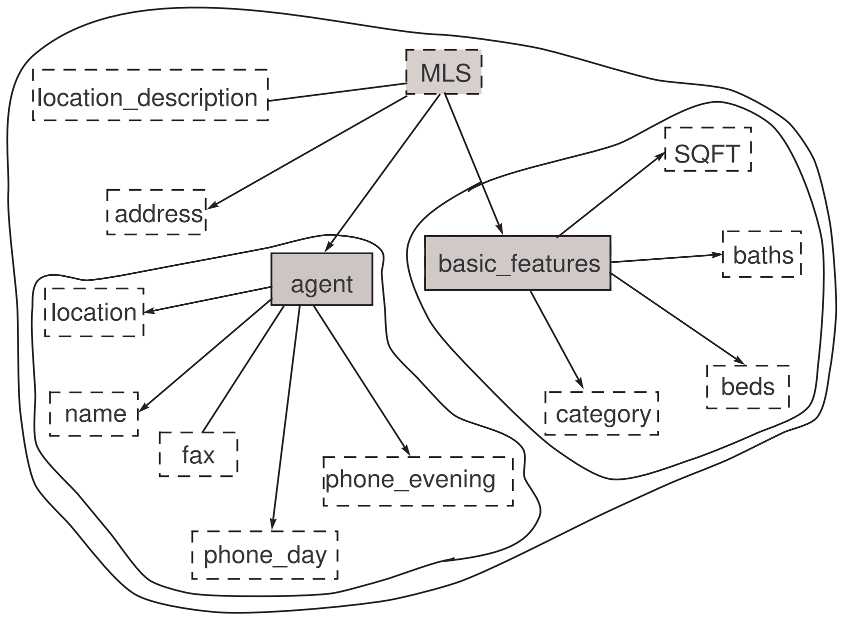
Abstract components in target and source schemas make use of equivalence classes of object sets [20]. Based on the relationship-set constraints in a conceptual schema H, we partition the object sets of H into equivalence classes in which the objects of the object sets of an equivalence class are in a one-to-one correspondence. Let X and Y be subsets of the object sets in H, and let F be a set of functional dependencies over H

as denoted by the functional edges in H. The two subsets X and Y are *equivalent* if $X \rightarrow Y \in F^+$ and $Y \rightarrow X \in F^+$. If $X \rightarrow Y$ and $Y \rightarrow X$, we write $X \leftrightarrow Y$. The relation $\leftrightarrow$ over subsets of the set of object sets O_H is an equivalence relation because the relation is reflexive, symmetric, and transitive. For any equivalence relation formed from the relation $\leftrightarrow$, we can form a set of pairwise nonintersecting sets of object sets, where each set of object sets determines every other set functionally. An equivalence class is *trivial* if it only contains a single object set. Otherwise, the equivalence class is *nontrivial*. In Schema 2 of Figure 3.1(b), one non-trivial equivalence class is $\{House,\ MLS\}$. The other equivalence classes in Schema 2 and all the equivalence classes in Schema 1 are trivial.

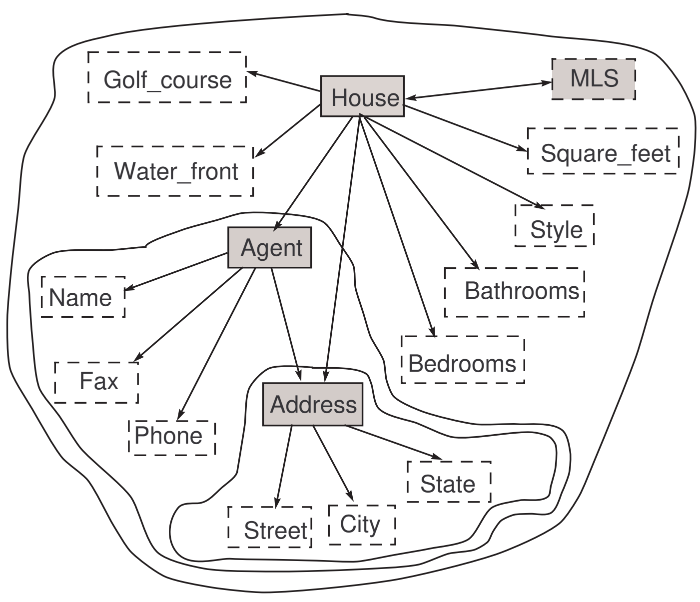
We further analyze equivalence classes in a conceptual schema H and divide them into a set of representative equivalence classes, which we denote as E_H^R, and a set of nonrepresentative equivalence classes. Intuitively, the set of representative equivalence classes of a conceptual schema H consists of those equivalence classes that are most important and informative for H. Formally, $e \in E_H^R$ is a *representative equivalence class* if and only if (1) the equivalence class e is nontrivial, or (2) the object set in e determines other set of object sets in H. In Figure 3.1, the representative equivalence classes in Schema 1 of Figure 3.1(a) are $\{MLS\}$, $\{agent\}$, and $\{basic\_features\}$ and the representative equivalence classes in Schema 2 of Figure 3.1(b) are $\{House,\ MLS\}$, $\{Agent\}$, and $\{Address\}$.

In addition to representative equivalence classes, the structure-matching technique makes use of one other notion, "context" of schema elements. By taking relationship sets around a representative equivalence class e into account, we cluster a set of object sets and relationship sets with a representative equivalence class as a conceptual component in the schema. We call this component the *context* for the representative equivalence class e, which we denote as $Cont_e$. The context $Cont_e$ for a representative equivalence class e consists a set of object sets, which we denote as $Cont_e^O$, and a set of relationship sets, which we denote as $Cont_e^R$, among the object sets in $Cont_e^O$.

We create the context in two phases. In the first phase, we construct a context beginning with each representative equivalence class e as follows. (1) Include all object sets in the functional closure h^+ of any element h of e. (2) Include all object sets adjacent



(a) Schema 1



(b) Schema 2

Figure 3.8: Context of construction for Schema 1 and Schema 2 after the first phase

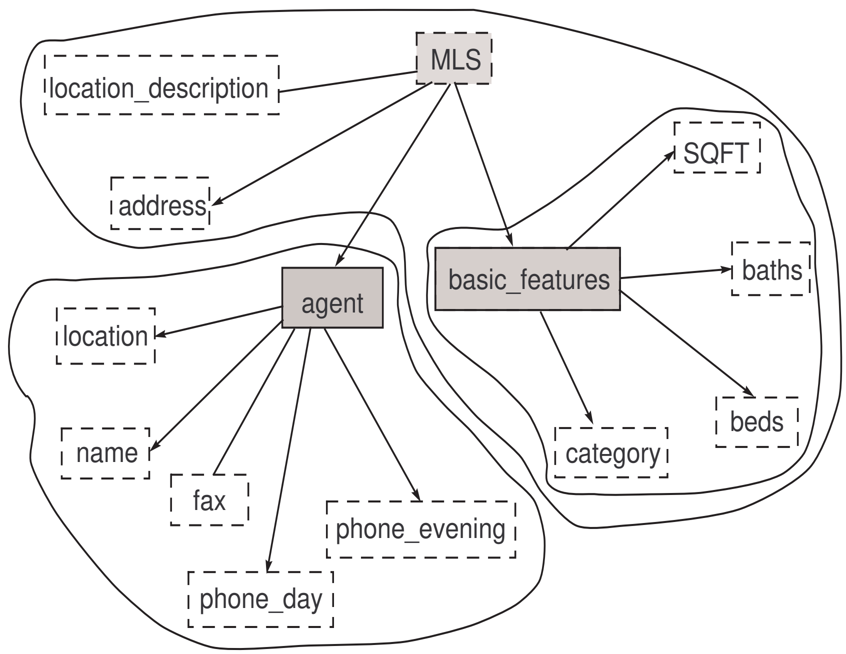
Input: a representative equivalence class e in a schema H.
Output: a context $Cont_e$ of e.

Include all the object sets in e into a set $Cont_e^O$
for each equivalence class e' connected by e
  if e' is non-representative
    Include the object set in e' into $Cont_e^O$
  else if e' is not compatible
    Include the object sets in the context of e' into $Cont_e^O$
Collect relationship sets among $Cont_e^O$ in H into a set $Cont_e^R$
Output the union of $Cont_e^O$ and $Cont_e^R$ as $Cont_e$

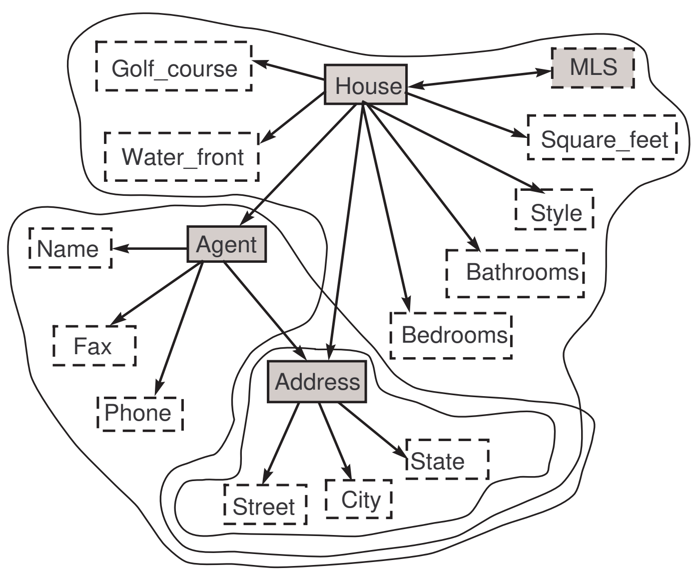
Figure 3.9: Context construction for representative equivalence classes

to object set in h^+ except object sets that belong to some other representative equivalence class. (3) Include all relationship sets that connect the included object sets. Figure 3.8 shows the construction after the first phase. In Figure 3.8, the representative equivalence-class elements are shaded and each enclosed area includes the object and relationship sets of the context for a representative equivalence class after the first phase. Consider, as an example, the first-phase construction of the context for MLS in Figure 3.8(a). The representative equivalence class e is $\{MLS\}$. The closure MLS^+ includes all the object sets in Figure 3.1(a) except $location\_description$ and fax. Since both $location\_description$ and fax, however, are adjacent to object sets included in MLS^+, we also include both $location\_description$ and fax in the first phase of construction of the context for MLS.

After the first phase of context construction for representative equivalence classes in target and source schemas, we compare the representative equivalence classes as well as their contexts between the schemas. In particular, we make a guess about how to map a source schema into a target schema by determining a set of *compatible* representative equivalence classes that are highly similar. In Figure 3.8, we guess that the representative equivalence class $\{House,\ MLS\}$ matches the representative equivalence class $\{MLS\}$, and likewise that $\{Agent\}$ matches with $\{agent\}$ because both the object sets and the contexts of the equivalence classes are similar. Then, in the second phase of context construction for representative equivalence classes, with the available guess about the map between the target and source schemas, we use the algorithm in Figure 3.9 to further construct the



(a) Schema 1



(b) Schema 2

Figure 3.10: Context of construction for Schema 1 and Schema 2 after the second phase

contexts for compatible representative equivalence classes in the target and source schemas. Figure 3.10 shows the new contexts for representative equivalence classes of schemas in Figure 3.1. Note that the contexts for the compatible representative equivalence classes $\{MLS\}$ and $\{House,\ MLS\}$ are smaller than their contexts in Figure 3.8 after the first phase. Essentially, the second phase separates the contexts of the elements into different contexts. This reduces the context scopes for representative equivalence classes and thus limits the search of finer-level correspondences to appropriate, smaller search spaces.

Our structure-matching technique discovers object-set matches as well as most relationship-set matches between contexts of compatible representative equivalence classes. We base our approach to structure matching for both direct and indirect matches on four intuitive ideas, which we illustrate using Schemas 1 and 2 in Figure 3.10.

1. *Nonlexical object sets*. Two nonlexical object sets match if their element names are similar and the lexical object sets around them describe similar data in the two schemas. A nonlexical object set has only object identifiers in a target or source schema. The object identifiers themselves do not describe the objects in the nonlexical object set. Instead, the values of object sets around the nonlexical object set describe nonlexical objects. The context analysis provides a limited scope for selecting the lexical object sets around the nonlexical object sets. In Figure 3.10, both $agent$ in Schema 1 and $Agent$ in Schema 2 represent the agent object for a house. The confidence $conf_1(agent,\ Agent)$, computed based on terminological relationships between the two element names, declares that $Agent$ of Schema 2 is similar to $agent$ of Schema 1 in Figure 3.10. The data associated with the adjacent $location$, $name$, fax, $phone\_day$, and $phone\_evening$ together represent the semantics of $agent$ in Schema 1. Similarly, the data associated with the adjacent $Name,\ Fax$, and $Phone$ as well as the adjacent $Street$, $City$, $State$ around the object $Address$ together represent the semantics for $Agent$ in Schema 2. By taking the adjacent lexical object sets into account, $agent$ of Schema 1 does match with $Agent$ of Schema 2.

2. *Equivalence classes*. The equivalence class $\{House,\ MLS\}$ of Schema 2 matches the equivalence class $\{MLS\}$ of Schema 1. With respect to the equivalence classes,

we can identify a match between the MLS object sets and we can see the similarity between lexical object sets that are closely related as already discussed in the above paragraph for nonlexical object-set matches. Assuming that Schema 1 is a source schema and Schema 2 is a target schema, we can create a virtual nonlexical object set $House'$ whose object identifiers are in a one-to-one correspondence with the values for MLS in Schema 1. The virtual nonlexical object set $House'$ matches with $House$ in Schema 2. Assuming, on the other hand, that Schema 2 is a source schema and Schema 1 is a target schema, we can simply match the MLS object sets directly.

3. *Lexical object sets*. Closely related sets of objects and values supply additional constraints for matching lexical object sets. In Figure 3.10(a), $address$ and $location$ denote house and agent locations respectively. Based on an analysis of the contexts for the two schemas, Figure 3.10(a) shows that the object set $location$ is in the context for $\{agent\}$ and $address$ is in the context of $\{MLS\}$. Thus, even though both $address$ and $location$ describe addresses, we distinguish the semantic correspondences in Figure 3.10(b) for $location$ and $address$ by considering context similarities. In Figure 3.10(b), the $Address$ objects are in both the context for $\{House,\ MLS\}$ and for $\{Agent\}$. With the compatibility between $\{House,\ MLS\}$ and $\{MLS\}$, given that the values for $Street$, $City$, and $State$ describe addresses in Figure 3.10(b), we can decide that there exists an indirect match between the addresses in the context of $\{MLS\}$ in Figure 3.10(a) and the addresses in the context of $\{House,\ MLS\}$ in Figure 3.10(b). Similarly, we can decide the other indirect match between the addresses in the context of $\{agent\}$ in Figure 3.10(a) and the addresses in the context of $\{Agent\}$ in Figure 3.10(b).

4. *Relationship sets*. Each relationship set, which is either in the source or in the target, is a graph, whose nodes and edges represent object sets and connections among the object sets respectively. A source relationship set s, which could be a virtual element, matches with a target relationship set t if and only if the two relationship sets s and t are isomorphic, i.e. a mapping function f_r from s to t exists such that there is an object set $f_r(o_s)$ with constraint c connected by t if and only if there is an object

set o_s with constraint c connected by s. Thus the requirement for relationship-set matches falls into two categories: (1) type requirements to satisfy node matching, and (2) constraint requirements to satisfy edge isomorphism. The type requirement between two nodes o_s and o_t is satisfied if and only if there exists an object-set match $o_t \sim o_s \Leftarrow \theta_{o_s}(\Sigma_S)$. To check constraint compatibility between two connections, [6] proposed four cases, which guide users' involvements for schema mapping operations while translating source data into the target. We adopt these same four cases to our work as follows. (1) The constraints on r_s and r_t are equivalent. Since this case satisfies the isomorphism constraints, nothing further need be done. (2) The constraints of r_s imply the constraints of r_t but not vice versa. In this case, since the relationships for r_s already necessarily satisfy the constraints of the target relationship set r_t, there is nothing further we need to do. (3) The constraints of r_t imply the constraints of r_s but not vice versa. In this case, we need to further transform r_s using *Selection* to restrict r_s to r'_s so that r'_s contains only relationships that satisfy the constraints of r_t. As a default, we can select relationships from r_s in an arbitrary order and keep only those that satisfy the constraints of r_t; otherwise a DBA must specify the selection criteria.[11] (4) Neither the constraints of r_t imply the constraints of r_s nor vice versa. This is a combination of (2) and (3) and thus, since there is nothing to do for (2), we can transform r_s as explained in (3).

Since our approach for schema mapping allows derived data in source schemas, an exhaustive search for relationship-set matches between Σ_T and V_S, where T is a target schema and S is a source schema, would have exponential time complexity. To avoid generating a large number of views over a source schema, we restrict the search space for view generation. Thus, our matching technique first discovers potential object-set matches. Then, with the guide of object-set correspondences, it discovers relationship-set matches.

Intuitively, we want to use the type requirements for relationship-set matches to trigger derivations of views over the source within a subset of a context, where the subset consists of a set of object sets and a set of relationship sets among the object sets. The

[11] See [6] for a detailed explanation of possibilities.

selection of the object sets is based on object-set matches and context restrictions. For example, let Schema 1 in Figure 3.10(a) be a target schema and Schema 2 in Figure 3.10(b) be a source schema, and assume that we discover that the objects in $Agent$ in the context of $\{Agent\}$ corresponds with the objects in $agent$ in the context of $\{agent\}$ and that the values for $Street$, $City$, and $State$ of objects $Address$ in the context of $\{Agent\}$ in Schema 2 semantically corresponds to $location$ in the context of $\{agent\}$ in Schema 1. To obtain the semantic correspondence of the relationship set $agent$—$location$ in Schema 1, with the available semantic correspondences between object sets in the contexts of $\{agent\}$ and $\{Agent\}$, we derive views over $Agent$—$Address$, $Address$—$Street$, $Address$—$City$, and $Address$—$State$ in the source in order to form a virtual relationship set that matches $agent$—$location$ in the target.

3.5 Mapping Algorithm

We have implemented an algorithm using our matching techniques that produces both direct and indirect matches between a source schema S and a target schema T. To illustrate our algorithm, we use our running example in Figure 3.1, and let Schema 1 be a source schema S and let Schema 2 be a target schema T.

Step 1: *Compute $conf$ measures between S and T.* For each pair of schema elements (s, t), which are either both lexical object sets or both nonlexical object sets, the algorithm computes a confidence value, $conf(s,t)$, as a combination of the output confidence values of the three nonstructural matching techniques as described in Section 3.3. We compute $conf(s,t)$ using the following formula:

$$conf(s,t) = \begin{cases} conf_1(s,t)\ , \text{ if } s \text{ and } t \text{ are nonlexical object sets} \\ 1.0\ , \text{ if } conf_3(s,t) = 1.0 \text{ and } s \text{ and } t \text{ are lexical object sets} \\ w_s(conf_1(s,t)) + w_v(conf^v(s,t))\ , \text{ otherwise} \end{cases}$$

	MLS	bath.	bed.	cat.	SQ.	loc._desc.	basic_feat.	agent	fax	ph._day	ph._even.	name	loc.	addr.
House	NA	NA	NA	NA	NA	NA	0.11	0.11	NA	NA	NA	NA	NA	NA
Bathrooms	0.14	0.92	0.66	0.09	0.27	0.07	NA	NA	0.14	0.14	0.14	0.07	0.14	0.14
Bedrooms	0.14	0.65	0.92	0.09	0.14	0.07	NA	NA	0.14	0.13	0.13	0.07	0.14	0.14
MLS	1.0	0.22	0.13	0.07	0.14	0.23	NA	NA	0.14	0.22	0.22	0.23	0.14	0.3
$Square_Feet$	0.14	0.27	0.14	0.07	1.0	0.07	NA	NA	0.44	0.22	0.22	0.08	0.14	0.14
$Water_Front$	0.07	0.07	0.07	0.14	0.07	1.0	NA	NA	0.07	0.07	0.07	0.14	0.07	0.08
$Golf_course$	0.07	0.07	0.07	0.14	0.07	1.0	NA	NA	0.07	0.07	0.07	0.14	0.07	0.07
Address	NA	NA	NA	NA	NA	NA	0.12	0.11	NA	NA	NA	NA	NA	NA
Agent	NA	NA	NA	NA	NA	NA	0.11	0.98	NA	NA	NA	NA	NA	NA
Fax	0.28	0.14	0.14	0.07	0.44	0.07	NA	NA	1.0	0.55	0.55	0.07	0.14	0.14
Phone	0.13	0.14	0.14	0.07	0.44	0.07	NA	NA	0.55	1.0	1.0	0.51	0.14	0.14
Name	0.23	0.07	0.07	0.27	0.08	0.43	NA	NA	0.07	0.52	0.52	1.0	0.08	0.09
Street	0.28	0.14	0.14	0.28	0.14	0.08	NA	NA	0.27	0.71	0.27	0.44	1.0	1.0
State	0.07	0.07	0.07	0.14	0.07	0.14	NA	NA	0.07	0.07	0.07	0.14	1.0	1.0
City	0.07	0.07	0.07	0.28	0.35	0.14	NA	NA	0.07	0.08	0.08	0.3	1.0	1.0
Style	0.28	0.07	0.07	0.66	0.23	0.66	NA	NA	0.23	0.23	0.23	0.27	0.07	0.51

Figure 3.11: Combined confidence-value matrix

In this formula, w_s and w_v are experimentally determined weights.[12] When the confidence value $conf_3(s,t)$ = 1.0, we let $conf_3$ dominate and assign $conf(s,t)$ as 1.0. The motivation for letting $conf_3(s,t)$ dominate is that when expected values appear in both source and target schema elements and they both match well with the values we expect, this is a strong indication that the elements should match (either directly or indirectly). Since the lightweight domain ontology is not guaranteed to be complete (and may even have some inaccuracies) for a particular application domain, the confidence values obtained from the other techniques can complement and compensate for the inadequacies of the domain knowledge. This motivates the third part of the computation for $conf(s,t)$, where $conf^v(s,t)$ either equals $(conf_2(s,t) + conf_3(s,t))/2$ if both the confidence values from data-value characteristics and expected-data values are available or equals $conf_2(s,t)$ if we cannot obtain $conf_3(s,t)$. Figure 3.11 shows the matrix that contains the combined confidence values obtained at the end of this step. An NA in the matrix denotes that we do not apply any confidence value between a lexical object set and a nonlexical object set in our approach for schema mapping.

[12]The two parameters w_s, which weights schema element names, and w_v, which weights schema element values, are application dependent. Using a heuristic guide, however, we can determine the two parameters based on schemas and available data even without experimental evidence. If the schema element names are informative and the data is not self descriptive, we assign w_s as 0.8 and w_v as 0.2. On the other hand, if the schema element names are not informative and the data is semantically rich, we assign w_s as 0.2 and w_v as 0.8. For all other cases, we assign both w_s and w_v as 0.5.

Step 2: *Analyze equivalence classes and their semantic correspondences between* S *and* T. Based on functional relationship sets, we identify two sets of equivalence classes, E_S in S and E_T in T. We next distinguish the representative equivalence classes in S and T as described in Section 3.4. Figure 3.8 shows these contexts after the first phase of context construction for representative equivalence classes of the schemas in Figure 3.1.

When comparing two representative equivalence classes $e_S \in E_S$ and $e_T \in E_T$ for the second phase of context construction, we take three factors into account: (1) the set of combined confidence measures $\{conf(s,t)|s \in e_S,\ t \in e_T\}$, (2) an importance similarity measure $sim_{importance}(e_S,\ e_T)$, and (3) a vicinity similarity measure $sim_{vicinity}(e_S,\ e_T)$. We can declare a compatible pair of representative equivalence classes, which we denote as $(e_T \sim e_S)$, if (1) one confidence value $conf(s',t') \in \{conf(s,t)|s \in e_S,\ t \in e_T\}$ is high, and (2) both the importance similarity $sim_{importance}(e_S,\ e_T)$ and the vicinity similarity measure $sim_{vicinity}(e_S,\ e_T)$ are high. The latter two measures together represent the similarity between contexts of e_S and e_T which we obtain after the first phase of context construction as discussed in Section 3.4. Given an experimentally determined threshold, th_{conf},[13] we calculate $sim_{vicinity}(e_S,\ e_T)$ and $sim_{importance}(e_S,\ e_T)$ based on the following formulas. In the formulas, $Cont_{e_S}$ ($Cont_{e_T}$) denotes the set of object and relationship sets for the context of e_S (e_T) and $Cont^O_{e_S}$ ($Cont^O_{e_T}$) denotes just the set of object sets in $Cont_{e_S}$ ($Cont_{e_T}$).

$$sim_{vicinity}(e_S,\ e_T) = max\left(\frac{|\{x|x \in Cont^O_{e_S} \wedge \exists y \in Cont^O_{e_T}(conf(x,y) > th_{conf})\}|}{|Cont^O_{e_S}|}, \frac{|\{x|x \in Cont^O_{e_T} \wedge \exists y \in Cont^O_{e_S}(conf(y,x) > th_{conf})\}|}{|Cont^O_{e_T}|} \right)$$

$$sim_{importance}(e_S,\ e_T) = 1.0 - \left|\frac{|Cont_{e_S}|}{|\Sigma_S|} - \frac{|Cont_{e_T}|}{|\Sigma_T|}\right|$$

Intuitively, $sim_{vicinity}$ measures the similarity of the vicinity surrounding e_S and the vicinity surrounding e_T, and $sim_{importance}$ measures the similarity of the "importance" of e_S

[13]For any application, the computed confidence values tend to converge to a specific high measure for element matches between two schemas. Thus we use a universal threshold value. Experimentally, we have determined that 0.8 works well across all applications.

and the "importance" of e_T where we measure the "importance" of an equivalence class e by counting the number of schema elements in the first-phase context of e. When the number of schema elements is largely different, [47] reports that it is difficult to determine the similarity based only on the singular measure, $sim_{vicinity}$. Thus, we add $sim_{importance}$, which is based on a conceptual-analysis technique discussed in [15] to help measure the context similarity from an additional perspective.

The comparison between equivalence classes in the target T and the source S provides the "guess" about semantic correspondences. Thus, by using the algorithm in Figure 3.9, we proceed with the second phase of context construction for representative equivalence classes in S and T. Figure 3.10 shows the modified contexts for representative equivalence classes in our running example.

At this point, we are finished with the top-level comparison between S and T. We are now ready to detect the object and relationship-set matches at the bottom-level.

Step 3: *Discover object- and relationship-set matches.* For each matching pair ($e_T \sim e_S$), which represents two compatible representative equivalence classes determined in Step 2, we use the combined confidence values between object sets to determine object-set matches. Figure 3.12 and Figure 3.13 show the confidence-value matrixes we base to settle object-set matches between the contexts for ($\{MLS\} \sim \{House, MLS\}$) and ($\{agent\} \sim \{Agent\}$) respectively. We first settle object-set matches between $Cont^O_{e_S}$ and $Cont^O_{e_T}$ that hold with the highest confidence value ($conf$ = 1.0). For all remaining unsettled object sets in $Cont^O_{e_S}$ and $Cont^O_{e_T}$, we find a best possible match using an injective-match settling algorithm so long as the confidence of the match is above the threshold, th_{conf}. As an example, after we obtain object-set matches between the contexts of $\{MLS\}$ and $\{House, MLS\}$ based on the perfect confidence values, we can further settle another two object-set matches by applying the injective-matching settling algorithm based on the high confidence values $conf(beds, Bedrooms)$ and $conf(baths, Bathrooms)$, which are above the threshold value th_{conf} = 0.8.

For each of the object-set matches, we keep the manipulation operations obtained by determining the expected-data-values patterns, which are required to transform source elements into virtual source object sets that directly match with target object sets.

	MLS	$baths$	$beds$	$category$	$SQFT$	$loc.\_desc.$	$basic\_feat.$	$address$
$House$	NA	NA	NA	NA	NA	NA	0.11	NA
$Bathrooms$	0.14	0.92	0.66	0.09	0.27	0.07	NA	0.14
$Bedrooms$	0.14	0.65	0.92	0.09	0.14	0.07	NA	0.14
MLS	1.0	0.22	0.13	0.07	0.14	0.23	NA	0.3
$Square\_feet$	0.14	0.27	0.14	0.07	1.0	0.07	NA	0.14
$Water\_front$	0.07	0.07	0.07	0.14	0.07	1.0	NA	0.08
$Golf\_course$	0.07	0.07	0.07	0.14	0.07	1.0	NA	0.07
$Address$	NA	NA	NA	NA	NA	NA	0.12	NA
$Street$	0.28	0.14	0.14	0.28	0.14	0.08	NA	1.0
$State$	0.07	0.07	0.07	0.14	0.07	0.14	NA	1.0
$City$	0.07	0.07	0.07	0.28	0.35	0.14	NA	1.0
$Style$	0.28	0.07	0.07	0.66	0.23	0.66	NA	0.51

Figure 3.12: Confidence-value matrix between contexts for $\{MLS\} \sim \{House, MLS\}$

	$agent$	fax	$phone\_day$	$phone\_evening$	$name$	$location$
$Address$	0.11	NA	NA	NA	NA	NA
$Agent$	0.98	NA	NA	NA	NA	NA
Fax	NA	1.0	0.55	0.55	0.07	0.14
$Phone$	NA	0.55	1.0	1.0	0.51	0.14
$Name$	NA	0.07	0.52	0.52	1.0	0.08
$Street$	NA	0.27	0.71	0.27	0.44	1.0
$State$	NA	0.07	0.07	0.07	0.14	1.0
$City$	NA	0.07	0.08	0.08	0.3	1.0

Figure 3.13: Confidence-value matrix between contexts for $\{agent\} \sim \{Agent\}$

For example, we keep the *Decomposition* operations identified by the expected-data-values patterns between $location$ in Schema 1 and $Street$, $City$, and $State$ in Schema 2. We will use these operations to specify mapping expressions for indirect matches in Steps 3 and 4.

With the available semantic correspondences between object sets in S and T, we further discover matches between relationship sets. We limit the recognition of most relationship-set matches to within the contexts of compatible representative equivalence classes between S and T. However, for relationship sets that go between the contexts of compatible representative equivalence classes, we identify semantic correspondences globally without the limitation of contexts. The recognition of a relationship-set match starts by locating a relationship set r_t in T. Then, based on the object sets O_{r_t} connected by r_t, we can locate a set of object sets that correspond to O_{r_t} in S, from which we either locate or derive a relationship set r_s that corresponds to r_t.

More particularly, between the contexts of two compatible representative equivalence classes, e_T in the target T and e_S in the source S, we first recognize semantic correspondences for relationship sets that connect object sets in e_T with relationship sets in or views over the context $Cont_{e_S}$ of e_S. As an example, given ($\{House, MLS\} \sim \{MLS\}$), we start processing the relationship set $House$—MLS in the context of $\{House, MLS\}$ in the target. To obtain its corresponding relationship set in the source, we use a ***Skolemization*** operator to derive a virtual relationship set $House'$—MLS in the context $\{MLS\}$. We next recognize semantic correspondences for target relationship sets each of which connects at least one object set that is in $Cont^{O}_{e_T}$ but not in e_T with relationship sets in or views over $Cont_{e_S}$. For example, to match with the target relationship set $House$—$Bedrooms$ in the context of $\{House,\ MLS\}$, which connects one object set $Bedrooms$ that is not in $\{House,\ MLS\}$, we use the ***Join*** and ***Projection*** operators to derive a virtual relationship set $House'$—$beds$ over the context of $\{MLS\}$ in the source.

After discovering relationship-set matches within contexts of compatible representative equivalence classes, we discover the semantic correspondences for target relationship sets that contains relationships connecting objects in different contexts of matching representative equivalence classes. In our running example, $House$—$Agent$ is such a relationship set in the target connecting object sets in the contexts of $\{House,\ MLS\}$ and $\{Agent\}$ in Figure 3.10(b). With the available object-set correspondence between $House'$ and $House$ and the correspondence between $agent$ and $Agent$, we derive a virtual relationship set $House'$—$agent$ in the source that corresponds $House$—$Agent$ in the target.

We now give all the derivation of all virtual object and relationship sets obtained in Step 3 for our running example. In the derivation, the assignment arrow ($\Leftarrow$) in each step denotes a virtual schema element on the left derived by applying the algebra expression on the right.

1. *Derivation of virtual object and relationship sets in the context of* $\{MLS\}$.

$$
\begin{aligned}
&House'\text{—}MLS \Leftarrow \varphi_{f_{House'}}(MLS)\\
&House' \Leftarrow \pi_{House'}(House'\text{—}MLS)\\
&House'\text{—}Address1' \Leftarrow \varphi_{f_{Address1'}}(House')\\
&Address1' \Leftarrow \pi_{Address1'}(House'\text{—}Address1')\\
&Address1'\text{—}address \Leftarrow \pi_{Address1',address}(MLS\text{—}House' \bowtie House'\text{—}Address1'\\
&\qquad\qquad \bowtie MLS\text{—}address)\\
&House'\text{—}baths \Leftarrow \pi_{House',baths}(MLS\text{—}basic\_features \bowtie basic\_features\text{—}baths\\
&\qquad\qquad \bowtie House'\text{—}MLS)\\
&House'\text{—}beds \Leftarrow \pi_{House',beds}(MLS\text{—}basic\_features \bowtie basic\_features\text{—}beds
\end{aligned}
$$

$$\bowtie House'\text{—}MLS)$$

$$House'\text{—}SQFT \Leftarrow \pi_{House',SQFT}(MLS\text{—}basic\_features \bowtie basic\_features\text{—}SQFT \bowtie House'\text{—}MLS)$$

$$House'\text{—}location\_description \Leftarrow \pi_{House',location\_description}(MLS\text{—}location\_description \bowtie House'\text{—}MLS)$$

These derivations are based on correspondences determined by using the confidence-value matrix in Figure 3.12 between object sets in the context of $\{MLS\}$ and object sets in the context of $\{House, MLS\}$. The structure-matching technique decides that the values for MLS, $baths$, $beds$, $SQFT$ in the context of $\{MLS\}$ of Figure 3.10(a) directly correspond to the values for MLS, $Bathrooms$, $Bedrooms$, $Square\_feet$ in the context of $\{House,\ MLS\}$ of Figure 3.10(b) respectively. Based on the available object-set correspondences, the algorithm derives $House'$—MLS, $House'$, $House'$—$baths$, $House'$—$beds$, $House'$—$SQFT$. In addition to the direct object-set matches, the algorithm determines that the values for $location\_description$ in the source are generalizations of the lot description implied in the values for $Water\_front$ and $Golf\_course$ in the target. Thus, the algorithm derives $House'$—$location\_description$ based on this indirect match as well as the newly obtained object-set match between $House'$ and $House$. Moreover, since the values for $Street$, $City$, and $State$ in the target are split values for values of $address$ in the source, the algorithm also derives $House$—$Address1'$, $Address1'$, and $Address1'$—$address$.

2. *Derivation of virtual object and relationship sets in the context of* $\{agent\}$.

$$agent\text{—}Address2' \Leftarrow \varphi_{f_{Address2'}}(agent)$$
$$Address2' \Leftarrow \pi_{Address2'}(agent\text{—}Address2')$$
$$Address2'\text{—}location \Leftarrow \pi_{Address2',location}(agent\text{—}Address2' \bowtie agent\text{—}location)$$
$$agent\text{—}'fax \Leftarrow \sigma_{key(agent)}(agent\text{—}fax)$$

Based on the confidence-value matrix in Figure 3.13, the structure-matching technique decides that the objects for $agent$ in the source directly correspond to the objects for $Agent$ in the target and that the values for $location$ in the source indirectly correspond to the values for $Street$, $City$, and $State,$ in the target. With these object-set correspondences, the algorithm derives the virtual elements $agent$—$Address2$, $Address2'$ and $Address2'$—$location$. In addition to the correspondence between $agent$ and $Agent$, the algorithm also determines the correspondence between fax and Fax because $conf(fax, Fax)$ = 1.0 in Figure 3.13. However, the relationships in $Agent$—Fax in the target are only a subset of the relationships in $agent$—fax in the source because the functional dependency $Agent \rightarrow Fax$ in the target more tightly constrains the relationship set than does the many-many relationship set in the source. As a default, we transform $agent$—fax with the *Selection* operator $\sigma_{key(agent)}$ which selects as many relationships as possible while maintaining the property that $agent$ is a key in the new relationship set and thus ensures that the FD $agent \rightarrow fax$ holds. To allow for an alternate to the default, the system alerts the DBA to the constraint violation and lets the DBA specify a different selection condition, if desired.

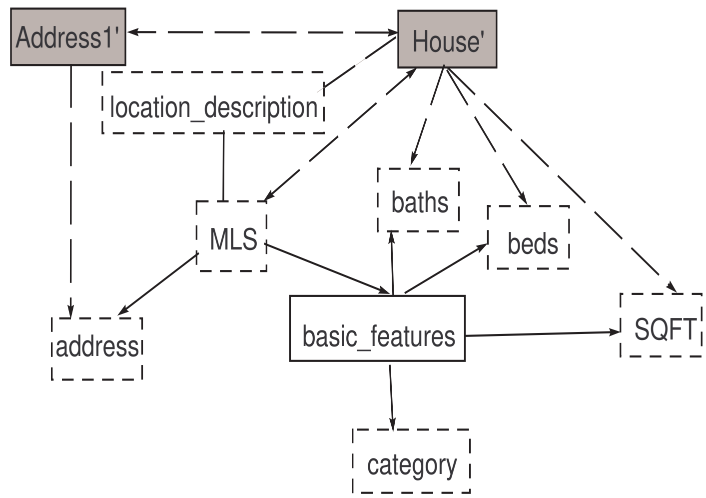
3. *Derivation of a virtual relationship set between the contexts of* $\{MLS\}$ *and* $\{agent\}$.

$$House'\text{—}agent \Leftarrow \pi_{House',agent}(House'\text{—}MLS \bowtie MLS\text{—}agent)$$

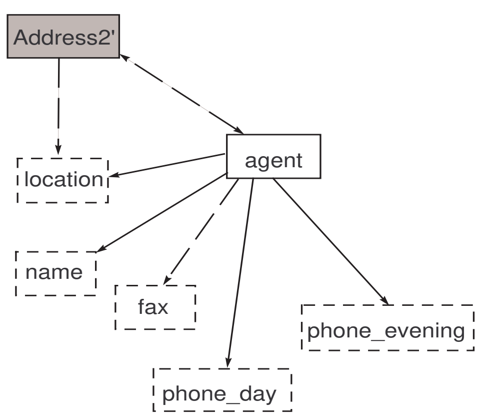
Note that even though the view derivation happens beyond the limitation of contexts of compatible representative equivalence classes, we still can constrict the search spaces by using available object- and relationship-set matches obtained within the two contexts.

Figure 3.14 shows the virtual object and relationship sets obtained when detecting relationship-set matches in Step 3. The dashed lines represent virtual relationship sets, and the shaded boxes represent virtual object sets.

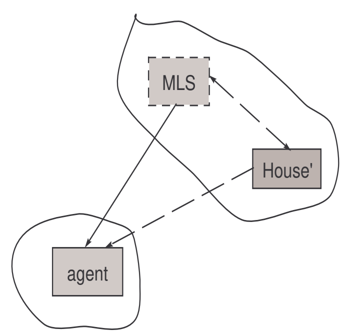
Step 4: *Specify mapping expressions for object- and relationship-set matches.* For direct matches, the specification of mapping expressions for mapping elements is straightforward. However, the specification of mapping expressions for indirect matches is nontrivial. Within this step, we use a bottom-up strategy to derive mapping expressions for indirect



(a) In the Context of $\{MLS\}$



(b) In the Context of $\{Agent\}$



(c) Inter Relationship Sets between the contexts of $\{Agent\}$ and $\{MLS\}$

Figure 3.14: Discovering object and relationship-set matches

matches. At the bottom level, we derive virtual elements based on instance-level information for indirect matches discovered in Step 3. Then, at the top level, we derive virtual elements based on schema-level information. We discuss the the two levels as follows.

1. *Instance-level derivations*. The derivation of virtual object and relationship sets by applying instance-level information depends on manipulation operations output from the matching technique while searching for expected-data values as described in Section 3.3. Figure 3.15 shows the virtual object and relationship sets derived after applying the instance-level information for our running example. Here, again, shaded boxes represent virtual object sets, and dashed lines to denote virtual relationship sets.

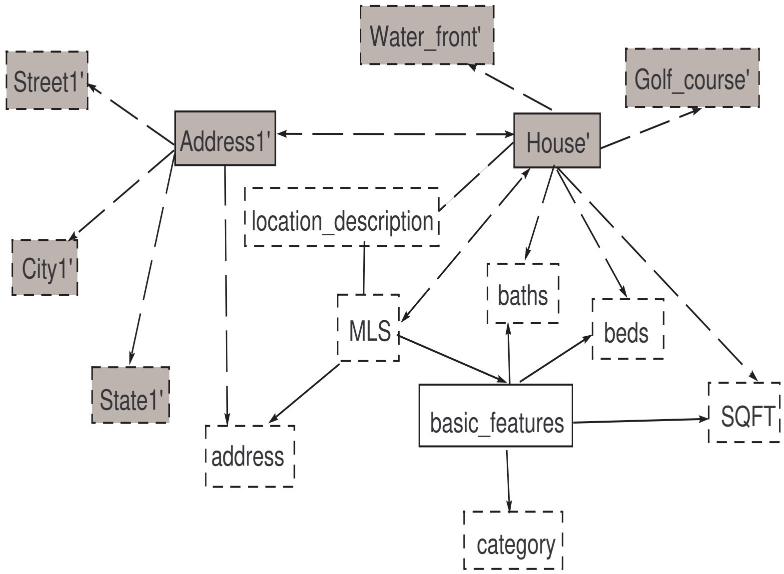
   - *Derivation of virtual object and relationship sets in the context of* $\{MLS\}$.

     $$House'\text{—}Golf\_course' \Leftarrow \beta^{\text{“}Yes\text{”},\text{“}No\text{”}}_{location\_description, Golf\_course'}(House'\text{—}location\_description)$$
     $$Golf\_course' \Leftarrow \pi_{Golf\_course'}(House\text{—}Golf\_course')$$
     $$House'\text{—}Water\_front' \Leftarrow \beta^{\text{“}Yes\text{”},\text{“}No\text{”}}_{location\_description, Water\_front'}(House'\text{—}location\_description)$$
     $$Water\_front' \Leftarrow \pi_{Water\_front'}(House\text{—}Water\_front')$$
     $$Address1'\text{—}Street1' \Leftarrow \pi_{Address1', Street1'}(\gamma^{R^{address}_{Street1'}}_{address, Street1'}(Address1'\text{—}address))$$
     $$Street1' \Leftarrow \pi_{Street1'}(Address1'\text{—}Street1')$$
     $$Address1'\text{—}City1' \Leftarrow \pi_{Address1', City1'}(\gamma^{R^{address}_{City1'}}_{address, City1'}(Address1'\text{—}address))$$
     $$City1' \Leftarrow \pi_{City1'}(Address1'\text{—}City1')$$
     $$Address1'\text{—}State1' \Leftarrow \pi_{Address1', State1'}(\gamma^{R^{address}_{State1'}}_{address, State1'}(Address1'\text{—}address))$$
     $$State1' \Leftarrow \pi_{State1'}(Address1'\text{—}State1')$$

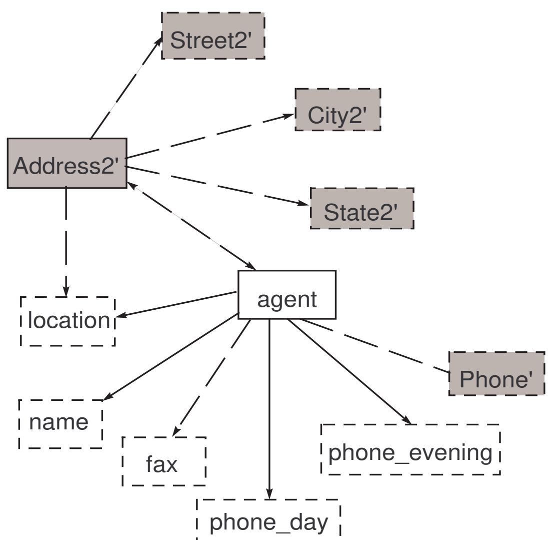
     The *DeBoolean* operators make new virtual relationship sets such that the values in the formed virtual relationship sets use Boolean indicators “Yes”/“No” as values. The three *Decomposition* operators use routines $R^{address}_{Street1'}$, $R^{address}_{City1'}$, and $R^{address}_{State1'}$ to decompose the string values for $address$ as values for the new virtual object sets $Street1'$, $City1'$, and $State1'$.

   - *Derivation of virtual object and relationship sets in the context of* $\{agent\}$.

     $$Address2'\text{—}Street2' \Leftarrow \pi_{Address2', Street2'}(\gamma^{R^{location}_{Street2'}}_{location, Street2'}(Address2'\text{—}location))$$
     $$Street2' \Leftarrow \pi_{Street2'}(Address2'\text{—}Street2')$$



(a) In the context of $\{MLS\}$



(b) In the context of $\{Agent\}$

Figure 3.15: Derived virtual object and relationship sets based on expected-data values

$$Address2'\text{—}City2' \Leftarrow \pi_{Address2',City2'}(\gamma^{R^{location}_{City2'}}_{location,City2'}(Address2'\text{—}location))$$
$$City2' \Leftarrow \pi_{City2'}(Address2'\text{—}City2')$$
$$Address2'\text{—}State2' \Leftarrow \pi_{Address2',State2'}(\gamma^{R^{location}_{State2'}}_{location,State2'}(Address2'\text{—}location))$$
$$State2' \Leftarrow \pi_{State2'}(Address2'\text{—}State2')$$
$$agent\text{—}Phone' \Leftarrow \rho_{phone\_day \leftarrow Phone'}(agent\text{—}phone\_day)$$
$$\cup \rho_{phone\_evening \leftarrow Phone'}(agent\text{—}phone\_evening)$$
$$Phone' \Leftarrow \pi_{Phone'}(agent\text{—}Phone')$$

The three *Decomposition* operators make virtual relationship sets based on routines $R^{location}_{Street2'}$, $R^{location}_{City2'}$, and $R^{location}_{State2'}$, which decompose the string values for $location$ as values for the new virtual object sets $Street2'$, $City2'$, and $State2'$. Indeed, the three routines used here are the same as those used to extract values for $Street1'$, $City1'$, and $State1'$ in the context of $\{MLS\}$. The agent's $Phone'$ values are a union of the $phone\_day$ and $phone\_evening$ values.

- *Derivation of virtual object and relationship sets between the contexts of* $\{MLS\}$ *and* $\{agent\}$. The indirect match between $House'\text{—}agent$ in the source and $House\text{—}Agent$ in the target does not depend on any manipulation operation derived by applying expected-data values. Thus we do not need to to derive virtual elements between the contexts of $\{MLS\}$ and $\{agent\}$.

2. *Schema-level derivations*. The matching techniques apply source and target schema structural characteristics to derive virtual object and relationship sets beyond the constraints of contexts. Basically, we collect matches that occur in different context pairs. For example, both objects in $Address1'$ in the context of $\{MLS\}$ and objects in $Address2'$ in the context of $\{agent\}$ in the source correspond to objects in $Address$ in the target, which is in both the context of $\{House,\ MLS\}$ and $\{Agent\}$. In a source-to-target mapping, between S and T, however, a target element $t \in \Sigma_T$ corresponds to at most one source element $s \in V_S$. Thus we use *Union* or *Selection* operations to force the one-to-one relationship sets between source elements in V_S and target elements in Σ_T.

   - *Derivation of virtual object sets for indirect object-set matches.*

$$Address' \Leftarrow \rho_{Address1' \leftarrow Address'} Address1' \cup \rho_{Address2' \leftarrow Address'} Address2'$$
$$Street' \Leftarrow \rho_{Street1' \leftarrow Street'} Street1' \cup \rho_{Street2' \leftarrow Street'} Street2'$$
$$City' \Leftarrow \rho_{City1' \leftarrow City'} City1' \cup \rho_{City2' \leftarrow City'} City2'$$
$$State' \Leftarrow \rho_{State1' \leftarrow State'} State1' \cup \rho_{State2' \leftarrow State'} State2'$$

It is an object-identity problem to merge any objects in $Address1'$ and $Address2'$ as objects in $Address'$. Recognizing object identity is beyond the scope of this chapter, we thus assume that we have a resolution or that duplicates do not matter. If the *Selection* operator is needed (*Selection* is not needed when Schema 2 is the target, as we currently are assuming), we may be able to recognize keywords for values in an object set to sort out the specializations. If not, a human expert may not be able to sort these out either.

- *Derivation of virtual relationship sets for indirect relationship-set matches.*

$$\begin{aligned}
Address'\text{—}Street' \Leftarrow\ & \rho_{Address1' \leftarrow Address', Street1' \leftarrow Street'} Address1'\text{—}Street1 \\
& \cup \rho_{Address2' \leftarrow Address', Street2' \leftarrow Street'} Address2'\text{—}Street2' \\
Address'\text{—}City' \Leftarrow\ & \rho_{Address1' \leftarrow Address', City1' \leftarrow City'} Address1'\text{—}City1 \\
& \cup \rho_{Address2' \leftarrow Address', City2' \leftarrow City'} Address2'\text{—}City2' \\
Address'\text{—}State' \Leftarrow\ & \rho_{Address1' \leftarrow Address', State1' \leftarrow State'} Address1'\text{—}State1 \\
& \cup \rho_{Address2' \leftarrow Address', State2' \leftarrow State'} Address2'\text{—}State2' \\
agent\text{—}'Phone' \Leftarrow\ & \sigma_{key(agent)}(agent\text{—}Phone')
\end{aligned}$$

The last derivation forces the participation constraint of $agent$ in the relationship set $agent\text{—}'Phone'$ to match with the functional constraint of $Agent$ in the relationship set $Agent\text{—}Phone$.

Figure 3.16 shows the source elements, which are object- and relationship-sets outside of the enclosed area, in the source-to-target mapping between S and T of our running example. The open white triangle denotes generalization/specialization. We use this notation to illustrate that the objects in $Address'$ are union of the objects in $Address1'$ and $Address2'$. Note that the object set $Address$ in the target directly matches the virtual object set $Address'$ in the source. The relationship sets $House\text{—}Address$ and $Agent\text{—}Address$ in the target, however, match with

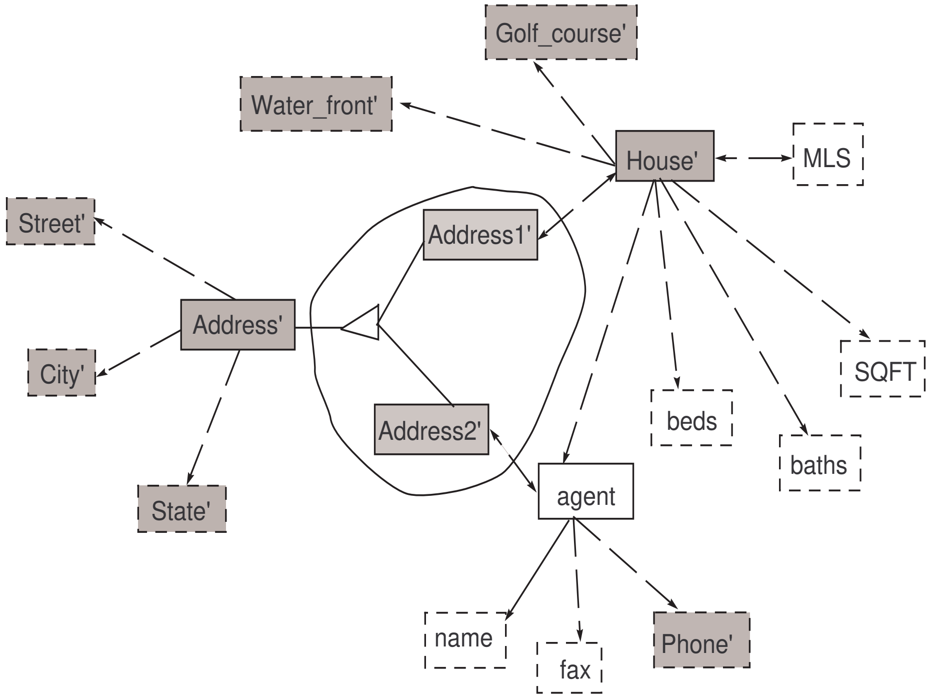


Figure 3.16: Source elements in the source-to-target mapping between Schema 1 (source) and Schema 2 (target)

$House'$—$Address1'$ and $agent$—$Address2'$ respectively. Thus both the object set $Address'$ and the relationship sets $House'$—$Address1'$ and $agent$—$Address2'$ are in the source-to-target mapping but neither $Address1'$ nor $Address2'$ is in the mapping.

Step 5: *Output a source-to-target mapping*. All the derivation in Step 3 and 4 generates the virtual object and relationship sets for the source-to-target mapping for our running example. At this point, there is a one-to-one mapping between source and target object and relationship sets. Thus, for example, $Address'$ maps $Address$, $House'$—$Address1'$ maps to $House$—$Address$, and $agent$—$'fax$ maps to $Agent$—Fax.

3.6 Data Integration

TIQS operates in two phases: design and query processing. We have just completed our discussion of the design phase, and we are now ready to discuss the query-processing phase. In the design phase, the system synergistically automates the generation of source-to-target mappings. For a mapping element $t \sim s \Leftarrow \theta_s(\Sigma_S)$, we think of the source element s is a *virtual target-view element* for t. This leads automatically to a rewriting of every target element as a union of corresponding virtual target-view elements. In the query processing phase, a user poses queries in terms of target relations. Query reformulation thus reduces to rule unfolding by applying the view-definition expressions for the target relations in the same way that database systems apply view definitions.

3.6.1 The Data Integration System

Definition 1. A *data-integration system I* is a triple (T, $\{S_i\}$, $\{M_i\}$), where T is a target schema, $\{S_i\}$ is a set of n source schemas, and $\{M_i\}$ is a set of n source-to-target mappings, such that for each source schema S_i there is a mapping M_i from S_i to T, $1 \leq i \leq n$.

To specify the semantics of I, we start with a *valid interpretation* D_{S_i} of each source schema $S_i \in \{S_i\} \in I$, $1 \leq i \leq n$. For an interpretation of a schema H to be valid, each tuple in D_H must satisfy the constraints specified for H. In our running example, let S_1 denote the information source whose schema is Schema 1 in Figure 3.1(a). For purposes of illustration, assume that we have one additional information source, which we denote as S_2, and assume that the wrapped Schema for S_2 is exactly the same as Schema 2 in Figure 3.1(b). Further, assume that both S_1 and S_2 have valid interpretations. The tables in Figure 3.17 show some partial populated data for relations in S_1 and S_2.

A target interpretation D_{S_iT} with respect to a source interpretation D_{S_i} in I (1) is a valid interpretation of T, and (2) satisfies the mapping M_i between S_i and T with respect to D_{S_i}. Assume that the mapping function for M_i is f_i. If f_i matches s_k with t_j, c is a tuple for t_j in D_{S_iT} if and only if c is a tuple for s_k derived through applying the mapping expression $\theta_{s_k}(\Sigma_{S_i})$ over D_{S_i}. For our running example, recall that the target schema is Schema 2 in Figure 3.1(b). Section 3.5 describes the source-to-target mapping between

MLS	location_description
M10	Water_front
M10	Golf_course
M11	Golf_course

MLS	basic_features
M10	b10
M11	b11

basic_features	SQFT
b10	1000
b11	1500

(a) Partial populated data for relations in S_1

House	Water_front
h1	Yes
h2	No
h3	Yes

House	Golf_course
h1	Yes
h2	Yes
h3	No

House	Square_feet
h1	990
h2	1420
h3	2500

(b) Partial populated data for relations in S_2

Figure 3.17: Some partial populated data for relations in sources

the target schema and the source schema for S_1 in Figure 3.1(a). Based on the source-to-target mapping, a valid target interpretation D_{S_1T} with respect to a valid interpretation D_{S_1} contains tuples in a valid interpretation for the schema in Figure 3.16. Moreover, since we assume that the source schema for S_2 is exactly the same as the target schema T, the source-to-target mapping between the target schema and the source schema S_2 is the identity mapping. Thus the tuples in a valid target interpretation D_{S_2T} with respect to D_{S_2} are those of source relations in S_2.

The semantics of I, which we denote as $sem(I)$, are defined as follows: $sem(I) = \{D_{S_iT} \mid D_{S_iT}$ is a target interpretation with respect to D_{S_i}, $S_i \in I\}$.[14] Intuitively, the semantics of I represent relevant data allowed in a predefined target schema T retrieved from available heterogeneous information sources. It is possible to prove that if a source has a valid interpretation, then the system can "load" data from the source into the target such that the part of the target populated from the source will necessarily have a valid interpretation [6].

[14]When sources share objects, both the object-identification problem and the data-merge problem need a resolution. (Note that neither this chapter nor other literatures that focus on data integration with virtual global schemas resolve these problems. The focus of this chapter is on mediation, mappings, and query reformulation.)

3.6.2 Query Reformulation

In the design phase, the data-integration system I collects the information that includes a target schema T, a set of source schemas $\{S_i\}$, and a set of source-to-target mappings $\{M_i\}$. In the query-processing phase, the system reformulates user queries in polynomial time.

In this dissertation, a query can be a conjunctive query, a conjunctive query with arithmetic comparisons, or a recursive query. We use the logic-rule notation in [70] to express user queries. Here, the queries are in terms of elements in Σ_T, which means that a predicate in a query body is either a target relation in Σ_T or a head predicate of a logic rule or an arithmetic comparison. We call a predicate that represents a target relation appearing in a query body a *target predicate*. As with pure DataLog, we do not allow negations of predicates that appear in user queries because we adopt the "Open World Assumption" in our approach. In our running example, assume that a user wants an answer to the query "For houses on water-front and golf-course property, list the number of square feet." We can express this query, which we denote as $q_{example}$, using the following logic rule:

$$\begin{aligned} House—Square\_feet(x, y) \ :- \ & House—Square\_feet(x, z), \\ & House—Golf\_course(x, \text{“}Yes\text{”}), \\ & House—Water\_front(x, \text{“}Yes\text{”}). \end{aligned}$$

Let q be a user query such as the one above. When evaluating q over $sem(I)$, the system I transparently reformulates q as q^{Ext}, which is a query evaluated by retrieving data from the underlying information sources in I. Let $D = \{D_{S_i} | S_i \in \{S_i\} \in I\}$ be the set of valid interpretations of source schemas in I. By reformulating q as q^{Ext}, the system transforms the task of evaluating q over $sem(I)$ into a task of answering q^{Ext} over D.

The system I reformulates a user query q by applying the inclusion dependencies for target relations collected in the design phase. Since a user poses queries in terms of elements in Σ_T, each target relation r_i that appears in the body of q corresponds to a set of inclusion dependencies ID_i, $1 \leq i \leq N$ and $N = |\Sigma_T|$. We expand the user query q for each inclusion dependency ($S_j.s_k \subseteq r_i$) in ID_i, where $s_k \in V_{S_j}$ ($1 \leq j \leq n$ and n is the number of sources), and where r_i is a target predicate that appears in the body of q, by adding a logic rule, $r_i(\overline{X}) \ :- \ S_j.s_k(\overline{X})$, where $\overline{X}$ is a vector of variables. Thus the added logic rules as well as the logic rule of q together form the query q^{Ext}.

To reformulate $q_{example}$ for our running example, the system I applies inclusion dependencies for target relations $House—Square\_feet$, $House—Water\_front$ and $House—Golf\_course$ appearing in the body of query $q_{example}$. In addition to the logic rule above for the user query $q_{example}$, the following logic rules are added to form the reformulated query $q^{Ext}_{example}$.

$$
\begin{aligned}
&House—Square\_feet(x,y) \ :- \ S_1.House'—SQFT(x,y)\\
&House—Square\_feet(x,y) \ :- \ S_2.House—Square\_feet(x,y)\\
&House—Water\_front(x,u) \ :- \ S_1.House'—Water\_front'(x,u)\\
&House—Water\_front(x,u) \ :- \ S_2.House—Water\_front(x,u)\\
&House—Golf\_course(x,v) \ :- \ S_1.House'—Golf\_course'(x,v)\\
&House—Golf\_course(x,v) \ :- \ S_2.House—Golf\_course(x,v)
\end{aligned}
$$

To evaluate a reformulated query q^{Ext} over sources, the system I decomposes q^{Ext} into subqueries, and retrieves and combines query answers to subqueries from individual information sources. We use a logic program P^{Ext}_D to describe the evaluation of q^{Ext} over D, where D is the set of valid interpretations of source schemas in I. The logic program P^{Ext}_D is defined as follows:

- *Rules*. The logic rules for q^{Ext}.

- *Facts*. For each source relation $S_j.s_k$ in the body of the logic program for q^{Ext}, we treat data for the source relations as ground facts. For example, if a tuple t is in the source relation $S_j.s_k$, we have the fact[15]: $S_j.s_k(t)$.

By evaluating P^{Ext}_D, the facts for the head predicate of q are query answers to the reformulated query q^{Ext} over D, which we denote as q^{Ext}_D. Note that when sending a sub-query to obtain data from an information source S_j, the system I also sends the mapping expression $\theta_{s_k}(\Sigma_{S_j})$ to the source S_j so that the source S_j correctly executes the mapping expression to derive source facts for s_k. Given the data in Figure 3.17, for our running query example,

[15] We use this logic program to capture the semantics of q^{Ext}_D. In real-world applications, query processing in I can optimize the evaluation of q^{Ext}.

$q_{example}$, we add the following list of source facts from S_1 and S_2 to the rules in $q^{Ext}_{example}$ above to form a logic program $P^{Ext}_{exampleD}$.

$S_1.House'$—$Water\_front'(h10, \text{"}Yes\text{"})$
$S_1.House'$—$Golf\_course'(h10, \text{"}Yes\text{"})$
$S_1.House'$—$Golf\_course'(h11, \text{"}Yes\text{"})$
$S_1.House'$—$Square\_feet(h10, 1000)$
$S_1.House'$—$Square\_feet(h11, 1500)$
$S_2.House$—$Water\_front(h1, \text{"}Yes\text{"})$
$S_2.House$—$Water\_front(h2, \text{"}No\text{"})$
$S_2.House$—$Water\_front(h3, \text{"}Yes\text{"})$
$S_2.House$—$Golf\_course(h1, \text{"}Yes\text{"})$
$S_2.House$—$Golf\_course(h2, \text{"}Yes\text{"})$
$S_2.House$—$Golf\_course(h3, \text{"}No\text{"})$
$S_2.House$—$Square\_feet(h1, 990)$
$S_2.House$—$Square\_feet(h2, 1420)$
$S_2.House$—$Square\_feet(h3, 2500)$

Note that the facts for S_1 have been transformed according to the source-to-target mapping in Section 3.5. For example, the object identifiers for house properties such as $h10$ are temporary object identifiers calculated through applying the *Skolemization* operators over the MLS numbers such as $M10$. By evaluating this logic program $P^{Ext}_{exampleD}$, we obtain the $House$—$Square\_feet$ facts $(h1, 990)$ and $(h10, 1000)$ in $q^{Ext}_{exampleD}$.

With query reformulation in place, we can now prove that query answers to any query are *sound*—every answer to a user query is a fact according to the semantics of I—and *maximal*—the query answers contain all the facts the sources have to offer with respect to the facts allowed in the global target schema. Let q_I denote the query answers to a user query q over the semantic of I, $sem(I)$, which represents all data relevant to the target schema T from all information sources in I. The proofs are based on an observation that the semantics of q_I can be captured using a logic program P_I. The logic program P_I is defined as follows.

- *Rule*. A user query q in terms of target relations in T.

- *Facts*. For each tuple t for a target relation r in D_{S_jT}, where $D_{S_jT} \in sem(I)$ and $S_j \in \{S_j\} \in I$, we have the fact: $r(t)$. (Note that these facts include all facts the sources have for T.)

The facts for the head predicate of q by evaluating P_I are the query answers q_I. Based on the characteristics of the two logic programs P_D^{Ext} and P_I, we can now prove the following theorem.

Theorem. *Let $I = (T, \{S_i\}, \{M_i\})$ be a data-integration system. Let $D = \{D_{S_i} | S_i \in \{S_i\} \in I\}$ be the set of valid interpretations of source schemas in I. Let q_I be the query answers obtained by evaluating q over $sem(I)$ and let q_D^{Ext} be the query answers obtained by evaluating q^{Ext} over D. Given a user query q in terms of target relations, a tuple $a = < a_1, a_2, \ldots, a_M >$ where M is the number of variables and constants in the head predicate of q, is in q_I if and only if a is a tuple in q_D^{Ext}.*

Proof. Assume that we define two logic programs P_D^{Ext} and P_I based on the semantics of q_D^{Ext} and q_I respectively as we discussed in Section 3.6.2.

If. Assume that a tuple $a = < a_1, a_2, \ldots, a_M >$ is in q_D^{Ext} but not in q_I. Since a is in q_D^{Ext}, there exists a substitution ϑ, which binds variables in P_D^{Ext} with constants such that the evaluation of $P_D^{Ext}\vartheta$ yields the tuple a for the head predicate of q. By using a subset of substitution ϑ' of ϑ for variables in q while evaluating P_I, since a is not in q_I, there must exist at least one subgoal of q that is not satisfied in $P_I\vartheta'$. Based on the evaluation theory in [62], the subgoal could be a target predicate or an arithmetic comparison. We make an analysis for each possibility as follows.

- *Arithmetic comparison*. Assume that we have an arithmetic comparison, which is a subgoal of q, satisfied in $P_D^{Ext}\vartheta$ but not in $P_I\vartheta'$. Since we use the same bindings for variables of q in ϑ and ϑ', if the arithmetic comparison is satisfied in $P_D^{Ext}\vartheta$, it must also be satisfied in $P_I\vartheta'$. Thus the unsatisfied subgoal is not an arithmetic comparison.

- *Target predicate*. Assume that we have a target predicate r, which is a subgoal of q, satisfied in $P_D^{Ext}\vartheta$ but not in $P_I\vartheta'$. Based on the substitution ϑ, the subgoal r

is satisfied in P_D^{Ext} because $r(c)$ holds, where c is a vector of constants using the bindings in ϑ. Since $r(c)$ holds, there must exist a rule $r(c) \ :- \ S_j.s_k(c)$ and a fact $S_j.s_k(c)$, where $S_j \in \{S_i\} \in I$ and $s_k \in V_{S_j}$, in $P_D^{Ext}\vartheta$ such that $r(c)$ is derived from the rule and the fact. Since we define the rule, $r(c) \ :- \ S_j.s_k(c)$, based on an inclusion dependency $(S_j.s_k \subseteq r)$, there must exist a mapping element $(r \sim S_j.s_k \Leftarrow \theta_{s_k}(\Sigma_{S_j}))$ between S_j and T. Since s_k matches with r and c is a tuple of s_k in S_j, based on the semantics of D_{S_jT} in $sem(I)$, c is a tuple of r in D_{S_jT}. Then, based on the definition of P_I, there must exist a ground fact $r(c)$ in P_I. Since $r(c)$ is a ground fact in P_I, the subgoal $r(c)$ is satisfied in $P_I\vartheta'$. This is contrary to the assumption. Thus the unsatisfied subgoal in P_I must not be a target predicate.

By analyzing the two possibilities, we conclude that all the subgoals in q are satisfied while evaluating $P_I\vartheta'$ to obtain the tuple a as a fact for the head predicate of q. Thus, based on the semantics of P_I, a is a tuple in q_I. This is contrary to our assumption.

Only if. Assume that we have a tuple $a = < a_1, a_2, \ldots, a_M >$ in q_I but not in q_D^{Ext}. Since a is a tuple in q_I, there must exist a substitution ϑ for variables in q such that the evaluation of $P_I\vartheta$ outputs the tuple a as a fact for the head predicate of q. By using the same substitution ϑ for variables in q while evaluating P_D^{Ext}, since a is not in q_D^{Ext}, there must exist at least one subgoal in q that cannot be satisfied in $P_D^{Ext}\vartheta$. Based on the evaluation theory in [62], the subgoal could be a target predicate or an arithmetic comparison. We make an analysis for each possibility as follows.

- *Arithmetic comparison*. Assume that we have an arithmetic comparison, which is a subgoal of q, satisfied in $P_I\vartheta$ but not in $P_D^{Ext}\vartheta$. Since we use the same bindings for variables of q in ϑ, if the arithmetic comparison is satisfied in $P_I\vartheta$, it must also be satisfied in $P_D^{Ext}\vartheta$. Thus the unsatisfied subgoal is not an arithmetic comparison.

- *Target predicate*. Assume that we have a target predicate r, which is a subgoal of q, satisfied in $P_I\vartheta$ but not in $P_D^{Ext}\vartheta$. Since the subgoal r is satisfied in $P_I\vartheta$, $r(c)$ holds, where c is a vector of constants by using bindings in ϑ. Since $r(c)$ holds, based on the definition of P_I, c is a tuple of r in a target interpretation D_{S_jT} of $sem(I)$, where

$S_j \in \{S_j\} \in I$. Based on the semantics of D_{S_jT}, since c is a tuple of r in D_{S_jT}, there must exist a mapping element $(r \sim S_j.s_k \Leftarrow \theta_{s_k}(\Sigma_{S_j}))$ between S_j and T where c is a tuple of the source relation $S_j.s_k$. By applying the mapping element that matches $S_j.s_k$ with r, we can derive an inclusion dependency $(S_j.s_k \subseteq r)$. Hence, since we have the inclusion dependency $(S_j.s_k \subseteq r)$ and the fact that c is a tuple of the source relation $S_j.s_k$, based on the definition of P_D^{Ext}, there must exist a rule $r(\overline{X}) :- S_j.s_k(\overline{X})$ and a fact $S_j.s_k(c)$ in $P_D^{Ext}\vartheta$, where $S_j \in \{S_j\} \in I$, $s_k \in V_{S_j}$, and $\overline{X}$ is a vector of variables corresponding attributes of r. Based on the rule and the fact, $r(c)$ holds in $P_D^{Ext}\vartheta$. This is contrary to the assumption. Thus the unsatisfied subgoal in $P_D^{Ext}\vartheta$ must not be a target predicate.

By analyzing the two possibilities, we conclude that all the subgoals in q are satisfied while evaluating $P_D^{Ext}\vartheta$ to obtain the tuple a as a fact for the head predicate of q. Thus, based on the semantics of P_D^{Ext}, a is a tuple in q_D^{Ext}. This is contrary to our assumption. $\Box$

3.7 Experimental Results

We evaluate the performance of our approach based on three measures: precision, recall and the F-measure, a standard measure for recall and precision together [2]. Given (1) the number of direct and indirect matches N determined by a human expert, (2) the number of correct direct and indirect matches C selected by our process described in this chapter and (3) the number of incorrect matches I selected by our process, we compute the recall ratio as $R = C/N$, the precision ratio as $P = C/(C + I)$, and the F-measure as $F = 2/(1/R + 1/P)$. We report all these values as percentages.

We tested the approach proposed here using the running example in our paper and also on several real-world schemas in three different application domains. In our experiments, we evaluated the contribution of different techniques and different combinations of techniques. We always used both structure and terminological relationships because given any two schemas, these techniques always apply even when no data is available. Thus we tested our approach with four runs on each source-target pair. In the first run, we considered only terminological relationships and structure. In the second run, we added data-value characteristics. In the third run, we replaced data-value characteristics with expected-data values, and in the fourth run we used all techniques together.

3.7.1 Running Examples

We applied the mapping algorithm explained in Section 3.5 to the schemas in Figure 3.1 populated (by hand) with actual data we found in some real-estate sites on the Web. First we let Schema 1 in Figure 3.1(a) be the source and Schema 2 in Figure 3.1(b) be the target. Then we reversed the schemas and let Schema 2 be the source and Schema 1 be the target.

Run Nr.	Number of Matches (N)	Number Correct (C)	Number Incorrect (I)	Recall %	Precision %	F-Measure %
1 (WS)	30	18	3	60%	86%	71%
2 (WCS)	30	18	1	60%	95%	73%
3 (WES)	30	30	0	100%	100%	100%
4 (WCES)	30	30	0	100%	100%	100%

W = Terminological Relationships using WordNet
C = Data-Value Characteristics
E = Expected Data Values
S = Structure

Table 3.1: Results for running example: source-Schema 1, target-Schema 2

Table 3.1 shows a summary of the results for each run in the first test where we let Schema 1 be the source and Schema 2 be the target. In the first run for the first test, the algorithm discovered eight direct matches correctly, but it also misclassified the source object set $address$ (meaning house address) and the virtual relationship set $house'$—$address$ by matching them with the target schema element $Style$ (meaning "apartment" or "townhouse") and $House$—$Style$. Also, the algorithm picked up a direct match between $phone\_day$ and $Phone$ but lost the correspondence between $phone\_evening$ and $Phone$. In the first run, the algorithm successfully discovered 10 of the 22 indirect matches. For example, by using the $Skolemization$ operator, the algorithm matches the object identifiers for a virtual nonlexical $House'$ based on the values in MLS with object identifiers in $House$. The mapping algorithm also correctly matches relationship sets, such as $House$—$Square\_feet$, $House$—$Bathrooms$, and $House$—$Bedrooms$, in the target with virtual relationship sets derived in the source based on *Join* and *Projection* operations. Especially, the algorithm uses the *Skolemization* operator two times to compute virtual objects in $Address1'$ and $Address2'$ that match with objects in $Address$ in the target, and correctly output a *Union* operation to union the two sets of object identifiers in a new virtual object set $Address'$ that directly matches with $Address$. In the second run, by adding the analysis of data-value characteristics, the two incorrect matches between $address$ and $Style$ and between $house'$—$address$ and $House$—$Style$ discovered based on terminological relationships disappeared, but the algorithm generated no more indirect matches than in the first run. In both the third and fourth runs, the algorithm successfully discovered all direct and indirect matches. Note that we correctly generated a $Selection$ operator to select the right subsets of $location\_description$ (meaning "view," etc.) in Schema 1 for $Water\_Front$ and $Golf\_Course$, and discarded the remaining values, which were inapplicable for Schema 2. The *Selection* operator sorted out values based on the expected-data values specified in the lightweight domain ontologies.

Run Nr.	Number of Matches (N)	Number Correct (C)	Number Incorrect (I)	Recall %	Precision %	F-Measure %
1 (WS)	25	15	3	60%	83%	70%
2 (WCS)	25	15	1	60%	94%	73%
3 (WES)	25	25	0	100%	100%	100%
4 (WCES)	25	25	0	100%	100%	100%

W = Terminological Relationships using WordNet
C = Data-Value Characteristics
E = Expected Data Values
S = Structure

Table 3.2: Results for running example: source-Schema 2, target-Schema 1

The result of the second test on our running example, in which we switched the schemas and let Schema 2 be the source schema and Schema 1 be the target schema, gave the results in Table 3.2. In the first run for the second test, the algorithm correctly discovered eight direct and seven of 17 indirect matches, but it also misclassified $Style$ and $House$—$Style$ by matching them with the target object set $address$ and a virtual relationship set $house'$—$address$. Because our approach used an injective-matching settling algorithm to obtain direct matches, the algorithm matched $Phone$ in the source with $phone\_day$ in the target but did not discover that the phones in $phone\_evening$ are also a subset of values in $Phone$. In the second run, by adding the analysis of data-value characteristics, the incorrect matches between $Style$ and $address$ and between MLS—$Style$, which is a virtual relationship set, and MLS—$address$ output based on terminological relationships disappeared. In both the third and fourth runs, the algorithm successfully discovered all direct and indirect matches. Especially noteworthy, we observed that our approach correctly discovered context-dependent indirect matches such as the semantic correspondence between $address$ in the target and $Street$, $City$, and $State$ in the target and appropriately produced operations consisting of a combination of ***Composition***, ***Join***, ***Projection***, and ***Selection***. The ***Selection*** operator sorted out the addresses composed from $Street$, $City$, and $State$ based on the two relationship sets $House$—$Address$ and $Agent$—$Location$ in Schema 2. Moreover, we correctly generated a ***Selection*** operator to specialize the $Phone$ value in Schema 2. The value transformation for ***Selection*** depends on keywords such as ***day-time***, ***day***, ***work phone***, ***evening***, and ***home*** associated with listed phone numbers. If the keywords are not available, however, the ***Selection*** operator fails to sort out the $Phone$ values.[16]

3.7.2 Real-World Examples

We considered three real-world applications: ***Course Schedule***, ***Faculty***, and ***Real Estate*** to evaluate our approach. We used a data set downloaded from the homepage of a schema-matching approach [19], Learning Source Descriptions (LSD), for these three applications, and we faithfully translated the schemas from DTDs used by LSD to rooted

[16]Even humans could not sort out this anomaly without the help of keywords.

conceptual-model graphs. Table 3.3 shows the characteristics of the source schemas. The table shows the number of object sets and relationship sets (Number of ObjSets and Number of RelSets), the maximum depth of the DTD trees. The rightmost column shows the percentage of object and relationship sets in a source schema that have either direct or indirect matches with other source schemas. The percentages show that the source schemas for *Course Schedule* and *Faculty* are relatively highly matchable; the source schemas for *Real Estate*, however, are not.

Domain	Number of Sources	Number of ObjSets	Number of RelSets	Depth	Matchable %
Course Schedule	5	15 - 19	14 - 18	1 - 4	62 - 93 %
Faculty	5	14	13	3	100%
Real Estate	5	34 - 88	33 - 86	1 - 4	17 - 73%

Table 3.3: Domains and schemas for real-world examples

For testing these real-world applications, we decided to let any one of the schema graphs for an application be the target and let any other schema graph for the same application be the source. We decided not to test any single schema as both a target and a source. Since for each application there were five schemas, we tested each application 20 times. Altogether we tested 60 target-source pairs. For each target-source pair, we made four runs, the same four (WS, WCS, WES, and $WCES$) we made for our running example. Altogether we processed 240 runs. Table 3.4 shows as summary of the results for the real-world data using all four techniques together.

In *Faculty* application, there were four indirect relationship-set matches because of constraints conflicts between relationship sets. Because the structure-matching technique correctly identified the constraints conflicts, for all four runs on *Faculty* every measure (recall, precision, F-measure) was 100%. Since the five source schemas are highly similar, but the data instances collected for each object sets are vastly different, we assigned a higher weight for w_S than w_V to dominate schema-level information.

In *Course Schedule* application, there were indirect relationship-set matches that required manipulations using *Join*, *Skolemization* and *Projection* operators. For *Course Schedule*, the first and second run achieved above 90% and below 95% on all measures; and the third and fourth run gave the results for *Course Schedule* as Table 3.4 shows. When using all four techniques, the correctly recognized mapping elements included 382 of 407 direct and 72 of 83 indirect matches. The incorrectly classified six mapping elements included four direct and two indirect matches. For direct matches, the precision, recall, and F-measures achieved 99%, 94%, and 96%; for indirect matches, the precision, recall, and F-measure achieved 98%, 88%, and 92%. Even when values for lexical object sets were not available in the first run, since most of the indirect matches appeared in this application are largely dependent on schema-level information, the mapping algorithm correctly identified 376 direct and 76 indirect matches for this application.

Application	Number of Matches (N)	Number Correct (C)	Number Incorrect (I)	Recall %	Precision %	F-Measure %
Course Schedule	490	454	6	93%	99%	96%
Faculty	540	540	0	100%	100%	100%
Real Estate	876	820	92	94%	90%	92%
All Applications	1906	1814	98	95%	95%	95%

Table 3.4: Results for real-world examples

The *Real Estate* application exhibited several indirect object- and relationship-set matches. The problem of *Merged/Split Values* appeared four times, the problem of *Subsets/Supersets* appeared 48 times, and the problem of *Object-Set Name as Value* appeared 10 times. The experiments showed that the application of expected-data values in the third and fourth run greatly affected the performance. In the first run, the performance reached 73% recall, 67% precision, and an F-measure of 70%. In the second run, the use of data-value characteristics improved the performance, but the measures were still below 80%. By applying expected-data values in the last two runs, however, the performance improved dramatically. The F-measures reached 91% in the third run and reached 92% by using all four techniques as Table 3.4 shows. The correctly classified 820 mapping elements included 417 of 453 direct matches and 403 of 423 indirect matches. The incorrectly classified 92 mapping elements included 24 direct and 68 indirect matches. Thus, for the direct matches, the precision, recall, and F-measure achieved 95%, 92%, and 93%; and for the indirect matches, the precision, recall, and F-measure achieved 86%, 95%, and 91%.

Our process successfully found all the indirect matches related to the problems of *Merged/Split Values* and *Object-Set Name as Value*. For the problem of *Subsets/Supersets*, our process correctly found all the indirect matches related to 44 of 48 problems of *Subsets/Supersets* and incorrectly declared four extra *Subsets/Supersets* problems. Of these eight, six of them were ambiguous, making it nearly impossible for a human to decide, let alone a machine. In four cases there were various kinds of phones for firms, agents, contacts, and phones with and without message features, and in another two cases there were various kinds of descriptions and comments about a house written in free-form text. The two clearly incorrect cases happened when the algorithm unioned (selected) office and cell phones and mapped them to phones for a firm instead of just mapping office phones to firm phones and discarding cell phones, which had no match at all in the other schema.

3.7.3 Discussion

The experimental results show that the combination of terminological relationships and structure alone can produce fairly reasonable results if schemas are highly matchable and indirect matches happen because of the problem *Path as Relationship Sets*. Moreover, the result shows that by adding our technique of using expected-data values, the performances are dramatically better even for applications, for example, *Real Estate*, whose schemas are relatively complex. Unexpectedly, the technique of using data-value characteristics did not help very much for these application domains. Our analysis of data-value characteristics is similar to the analysis in SEMINT [43], which produced good results for their test data. The data instances in the real-world applications we used, however, do not appear to be as regular as might be expected. The statistics are highly variant, for example, in applications such as *Course Schedule* and *Real Estate*. For these applications, a large amount of training data would be needed to train a universal decision tree required for this approach. Collecting enough training data, however, is not a trivial problem.

Some element matches failed in our approach partly because they are potentially ambiguous, and our assertions about what should and should not match are partly subjective.[17] Even though we tested our approach using the same test data set as in LSD [19], the answer keys were generated separately, and LSD focuses on computing direct object-set matches. Furthermore, neither the experimental methodologies nor the performance measures used are the same. With this understanding, we remark that they reported approximate accuracies of 70% for *Course Schedule*, 90% for *Faculty*, 70% and 80% for the two experiments they ran on the *Real Estate* application. Thus, although our raw performance numbers are an improvement over LSD [19], we do not try to draw any final conclusion.

One obvious limitation to our approach is the need to construct an application-specific domain ontology. Currently, we manually construct these domain ontologies. As we explained in Section 3.3, however, these domain ontologies are lightweight and are relatively easy to construct and need not be complete. It is possible, however, to make use of statistical learning techniques to collect a set of informative and representative keywords for application concepts in domain ontologies. Thus, without human interaction except for some labeling, we can make use of many keywords taken from the data of the application itself and thus specify regular-expression recognizers for the application concepts at least in a semi-automatic way. Furthermore, many values, such as dates, times, and currency amounts are common across many application domains and can easily be shared. Since domain ontologies appear to play an important role in indirect matching, finding ways to semi-automatically generate them is a goal worthy of some additional work.

One other limitation to our approach is that the schemas for real-world applications we used in the experiment are in same domains, even though they may not have significant overlap.[18] For example, the root nodes, which describe the designated object of

[17] It is not always easy to do ground-truthing [39].

[18] The matchable percentages in real estate applications range only from 17% to 73%

primary interest, of the five schemas used in each application are semantically correspondent. The structure matching we used applies a top-down strategy to compare two schemas. When discovering object- and relationship-set matches, the algorithm is robust enough to recognize the correspondences between object sets by applying available schema-level and data instance-level information. When discovering relationship-set matches, however, the algorithm is highly dependent on structure similarities during context-level comparison. As [19] mentioned, this is typically the case for "aggregator" domains, where the data-integration system provides access to sources that offer essentially the same service. In the future, we plan to examine more applications beyond such limitation using our algorithm.

3.8 Related Work

[59] provides a survey of several schema mapping systems, and [71] compares the two basic approaches GAV and LAV for data integration. We do not repeat this work here, but instead describe work related to our approach from three perspectives: (1) work on discovering direct matches for schema elements, (2) work on discovering indirect matches for schema elements, and (3) work on integrating data from heterogeneous information sources.

Direct Matches. Most of the approaches [5, 19, 23, 43, 47, 54, 57] to automating schema mapping focus only on generating direct matches for schema elements.

- In some of our previous work [23], we experimented using schema-level and instance-level information to help identify direct matches. In this dissertation, we extend this work to generate source-to-target mappings that contains both direct and indirect object- and relationship-set matches.

- As in our approach, the LSD system [19] applies a meta-learning strategy to compose several base matchers, which consider either data instances, or schema information. LSD largely exploits machine learning techniques. There are two phases in the LSD system: training and testing. In the training phase, LSD requires training data for each matchable element in a mediated schema for base matchers and the meta matcher. For each different application, however, both base and meta learners have to be supervised, and the supervisor must supply and mark training data to train the learners. Our approach differs in the three ways. (1) We applied machine learning algorithms only to terminological relationships and data-value characteristics. (2) Our system learned a universal decision tree for all application domains based on a domain-independent training set. Thus our system avoids the work of collecting and labeling training data for each application in LSD. (3) To combine techniques, we let structure features guide the matching based on the results from multiple kinds of independent matches. When porting to new application domains, we manually predefine domain ontologies, which can work for computations of source-to-target mappings between source and target schemas within domains. Especially, we can incrementally polish the domain ontologies for newly available schemas to improve mapping performances. LSD could improve its performance either by marking a large amount of domain-dependent training data or by supervising the training in a clever way using active-learning techniques. However, usually the sample data for each matching element of the mediated schema in multiple learners is not easy to collect and is tedious to label, and how to guide the learning in active learning for different learning models is not trivial either. Active learning may need many labeled examples to "bootstrap" learners so that it makes good estimates about which unlabeled examples are useful, and it also takes much effort to choose initial training examples.

- SEMINT [43] applies neural-network learning to automating schema mapping based on instance contents. It is an element-level schema matcher because it only considers attribute matching without taking the structure of schemas into account. It is flexible to port to new application domains because of the application of learning-based techniques as in LSD. However, SEMINT clusters attributes in one data source first, and then trains a classifier to classify each attribute in another source into a cluster of attributes, not a single attribute, in the first source. The tool reduces the search space for mapping elements although the result search space is still huge and requires human interaction to resolve the correct matches.

- The structure matching algorithm in Cupid [47] motivated our structure-matching technique. Cupid, however, does not properly handle two schemas that are largely different. Moreover, Cupid matches two schemas using a bottom-up strategy. Our mapping algorithm discovers direct and indirect matches using a top-down strategy.

- ARTEMIS [5], DIKE [57], and Cupid [47] exploit auxiliary information such as synonym dictionaries, thesauri, and glossaries. All their auxiliary information is schema-level—it does not consider data instances. In our approach, the auxiliary information, including data instances and domain ontologies, provides a more precise characterization of the actual contents of schema elements. The imported dictionary we use, WordNet, is readily available and no work is required to produce thesauri as in other approaches.

Indirect Matches. Some work on indirect matches is starting to appear [6, 47, 52, 55], but researchers are only beginning to scratch the surface of the multitude of problems.

- Both Cupid [47] and SKAT [55] can generate global $1 : n$ indirect matches [59]. To illustrate what this means, if in Figure 3.1 we let Schema 1 be the source and Schema 2 be the target, and if we make $Address$ a lexical object set rather than a nonlexical object set and discard $Street$, $City$, and $State$ in Schema 2, Cupid can match both $address$ and $location$ in the source directly with the modified $Address$ in the target. Thus Cupid can generate a global $1 : n$ indirect match through a *Union* operation. Our approach, however, can find indirect matches for $location$ and $address$ in the source with $Street$, $City$, and $State$ in the target based on finding expected-data values and using the *Decomposition* operator as well as the *Union* operator, something which is not considered in Cupid.

- The Clio project [52, 53] is a system for managing and facilitating the complex tasks of heterogeneous data transformation and integration. The objective of Clio is to support the generation and management of schemas, correspondences between schemas, and mappings (queries) between schemas. Clio has an extensive tool set to aid users semi-automatically generate mappings. The system introduces an interactive mapping creation paradigm based on value correspondence that shows how a value of a target schema element can be created from a set of values of source elements. A DBA, however, is responsible for inputting most of the value correspondences. Clio and TIQS are independently implemented. The mapping techniques in TIQS and Clio are complementary. The techniques of TIQS could help Clio discover both direct and indirect matches semi-automatically. On the other hand, TIQS could take advantage of the GUI provided by Clio such that a DBA can easily be involved in schema mapping. The components of either Clio and TIQS could be altered to support the alternative approach.

- [6] proposes a mapping generator to derive an injective target-to-source mapping including indirect matches in the context of information integration. The mapping generator raises specific issues for a user's consideration. The mapping generator, however, has not been implemented. Our work therefore builds on and is complementary to the work in [6].

Data Integration. Data integration is thought to be one of the most important problems in modern information systems. In this dissertation we focus on data-integration systems without materializing the global schemas. We describe several approaches related to the proposed approach for data integration in this dissertation.

- [12] surveys the most important query processing algorithms proposed in the literature for LAV [32, 42], and describes the principal GAV [17] data-integration systems and the form of query processing they adopt. In a GAV approach, query reformulation reduces to rule unfolding. However, changes in information sources or adding a new information source requires a DBA to revise the global schema and the mappings between the global schema and source schemas. Thus GAV is not scalable for large applications. LAV scales better and is easier to maintain than GAV because DBAs create a global schema independently of source schemas. Then, for a new (or changed) source schema, the DBA only has to give (adjust) a *source description* that describes source relations as views of the global schema. Automating query reformulation in LAV, however, has exponential time complexity with respect to query and source schema definitions. Thus LAV has low query performance when users frequently pose complex queries.

- [31] proposes a *Global-Local-as-View* (GLAV) approach, which combines expressive powers of both LAV and GAV. In a GLAV approach, the independence of a global schema, the maintenance to accommodate new sources, and the complexity to reformulate queries are the same as in LAV. However, instead of using a restricted form of first-order logical sentences as in LAV and GAV to define view definitions, GLAV uses flexible first-order sentences such that it allows a view over source relations to be a view over global relations in source descriptions. Thus GLAV can derive data using views over source relations, which is beyond the expressive ability of LAV, and it allows conjunctions of global relations, which is beyond the expressive ability of GAV. Our solution, TIQS, also has the ability to derive views over source schemas. The sets of view-creation operators in TIQS, however, are more powerful because GLAV has nothing comparable to merge/split or Boolean operators. Moreover, GLAV claims no ability to semi-automate the specification of source descriptions.

- [11] proposes a translation algorithm to turn LAV into GAV such that it can keep LAV's scalability and obtain GAV's simple query reformulation. The translation results in a logic program that can be used to answer queries using rule unfolding. However, even though the translation to obtain the logic program is in polynomial time, the evaluation of the logic program could produce an exponential number of facts because of recomputing source relations over all source data. In contrast, TIQS encapsulates views for source relations in mapping elements. Since the view definitions are immediately available, query processing in TIQS has better query performance than the translation approach. Furthermore, [11] does not claim the ability to semi-automate the specification of source descriptions.

3.9 Conclusions and Future Work

We presented a framework for automatically discovering both direct matches and many indirect matches between sets of source and target schema elements. In our framework, multiple techniques each contribute in a combined way to produce a final set of matches. Techniques considered include terminological relationships, data-value characteristics, expected values, and structural characteristics. We detected indirect element matches for *Join*, *Projection*, *Selection*, *Union*, *Skolemization*, *Composition*, and *Decomposition* operations as well as *Boolean* and *DeBoolean* conversions for *Object-Set Names as Value*. We base these operations and conversions mainly on expected values and structural characteristics. Additional indirect matches, such as arithmetic computations and value transformations, are for future work. We also plan to semi-automatically construct domain ontologies used for expected values, automate application-dependent parameter tuning, and test our approach in a broader set of real-world applications. As always, there is more work to do, but the results of our approach for both direct and indirect matching are encouraging, yielding about 90% in both recall and precision.

We formalized the output mapping between a source schema and a target schema from our matching framework as a source-to-target mapping. By applying source-to-target mapping based on a predefined target schema, we proposed TIQS, which provides solutions for five major issues: heterogeneity, scalability, continual infusion and change of local information sources, query processing complexity, and global-schema evolution in a unified approach to data integration. TIQS combines the advantages and avoids the limitations of both GAV and LAV. And it has polynomial-time query reformulation, and is easy to add or modify information sources. In summary, TIQS increases both scalability and usability as compared to previously proposed data-integration approaches.

Chapter 4

Conclusions

Recognizing applicable HTML documents and specifying source-to-target mappings are two critical steps for data-integration applications. Manual recognition and specification are expensive and error-prone. It is therefore important to develop techniques to automate these two steps for large-scale data-integration applications. In this dissertation, we proposed ways to automate applicability checking as well as the specification of source-to-target mappings. We enumerate the key contributions and discuss the direction of future work.

4.1 Contributions

The main contributions of this dissertation include one approach to recognize applicable HTML documents for a target application and another approach to exploit both schema-level and instance-level data and metadata to generate source-to-target mappings semiautomatically between target and source schemas. The following lists the contributions.

- *A formal statement of the applicability-checking problem.* We used an application ontology that specifies a particular application. An application ontology is able to explicitly specify the concepts as well as relationships among the concepts for an application. We introduced the notion of describing the heuristics that determine similarity between an HTML document and an application ontology. By rephrasing the applicability-checking problem as a classification problem, we were able to resolve this problem using machine learning techniques.

- *A robust framework to recognize applicable HTML documents for applications.* The framework exploits learning-based techniques to combine heuristics computed based on the specification of an application ontology. The framework is robust because it is able to incorporate more learning techniques to train learners for applications. Moreover, learners for applications can improve their performance incrementally by exploiting more existing knowledge such as more training data obtained as the system evolves.

- *A set of heuristics to recognize applicable HTML documents for applications.* For each application, we created a set of heuristics to measure the similarity between an HTML document and the application ontology, which we used to specify an application. The heuristics change as the application ontology changes because we compute the heuristics based on the extracted data obtained by applying the application ontology. When porting to a new application, the heuristics automatically adjust based on a new application ontology and given application data.

- *An empirical evaluation of the approach to recognize applicable HTML documents.* We processed two applications (car ads and obituaries) to evaluate the approach to the applicability-checking problem. The results are encouraging. The two learners we trained for both applications achieved above 95% for precision, recall, and F-measures.

- *A formal representation of a source-to-target mapping.* We extended the relational algebra such that we are able to formally define a source-to-target mapping to represent both direct and many indirect matches between a source schema and a target schema. The representation of a mapping element in a source-to-target mapping clearly declares both the semantic correspondences as well as the access paths to access and load data from sources into a target schema.

- *An extensible framework to discover mapping elements.* The framework we built to generate a source-to-target mapping is extensible. As new techniques become available to resolve the schema-mapping problem, the framework is able to import these new techniques to resolve the schema-mapping problem with the current techniques discussed in this dissertation.

- *An empirical evaluation of the approach to automate schema mapping.* We used a data set containing three real-world applications, *Course Schedule*, *Faculty*, and *Real Estate*, to evaluate the approach to schema mapping. For both direct and indirect matches, our approach yields about 90% for both recall and precision.

- *A formal definition of a data-integration approach.* We formally defined a data-integration approach, TIQS, based on source-to-target mappings. TIQS provided solutions for five major issues: heterogeneity, scalability, continual infusion and change of local information sources, query-processing complexity, and global-schema evolution in a unified approach to data integration.

- *A formal proof for query reformulation based on source-to-target mappings.* We proved that the query reformulation in TIQS reduces to rule unfolding based on source-to-target mappings. The reformulated queries extract all available answers from information sources with respect to the definition of TIQS.

4.2 Future Work

In the future, we would like to extend the research in this dissertation. The following lists the directions we would like to pursue in the future.

- *Testing more applications and data to evaluate the approaches.* We would like to evaluate the two approaches using a broad set of applications. In particular, we plan to test semistructured documents, such as XML documents, in addition to the HTML documents we considered in order to evaluate our approach to the applicability-checking problem.

- *Building graphical user interfaces for the two approaches.* For both the applicability-checking problem and the schema-mapping problem, our approaches required human's interactions to correctly recognize applicable HTML documents and to generate source-to-target mappings. A user-friendly environment is essential to improve human productivity. Thus we plan to build graphical user interfaces to facilitate the user's involvement.

- *Automating the ontological specifications for applications.* For the problem of applicability checking, the specification of application ontologies is essential to the success of our approach. For the problem of automating schema mapping, the specification of light-weight domain ontologies is one of the keys to discover indirect matches between target and source schemas. Manual specification of ontologies is expensive. Hence, we plan to use learning-based techniques to automate ontological specifications.

- *Exploiting the active-learning strategy to train learners.* When using C4.5 to train a learner for classifying applicable HTML documents for an application, a human expert semiautomatically selects training documents and labels training data at the training phase. We would like to further improve the training process. Hence, we plan to apply an active-learning strategy to automatically construct high-quality training data to train the learners.

- *Exploiting a meta-learning strategy to train learners.* The current approach for the applicability-checking problem exploited the decision-tree learning algorithm of C4.5 to train the learners for applications. There exist other approaches that have proven useful to resolve this problem [68, 73]. We therefore plan to extend the training classifier in our approach by exploiting multiple kinds of techniques. In order to combine the classification models output by multiple training classifiers, we plan to exploit a meta-learning strategy to combine the various classification models.

- *Analyzing and extending existing mapping specifications for data integration.* There are other expression languages to specify mappings between a global schema and source schemas in a data-integration approach. We plan to analyze the existing representation languages as well as the extended relational algebra we used to represent source-to-target mappings, choose appropriate notation, and extend the language to allow for additional kinds of indirect matches.

- *Automating mappings between ontologies on the Semantic Web.* The schema-mapping approach designed in this dissertation was for database-like conceptual-model instances of Web data. Establishing mappings between ontologies will be essential to the success of exchanging data on the Semantic Web. We plan to extend our approach to resolve the ontology-mapping problem and thus make a contribution to the future development of the Web.

- *Automating mappings between XML documents.* Similar to finding mappings between ontologies on the Semantic Web, we should be able to apply the technique described in this dissertation to find mappings between documents expressed in XML. This would be a way to help automate data exchange among E-business applications.

Bibliography

[1] Y. Arens, C.Y. Chee, C.-N. Hsu, and C.A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–159, June 1993.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, Menlo Park, California, 1999.

[3] L.D. Baker and A.K. McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21th ACM SIGIR (SIGIR'98)*, pages 96–103, Melbourne, Australia, August 24–28 1998.

[4] C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-based information mediation with MIX. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 597–599, Philadelphia, Pennsylvania, May 31–June 3 1999.

[5] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, March 1999.

[6] J. Biskup and D.W. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. *Information Systems*, 28(3):169–212, May 2003.

[7] Homepage for Bob Howard Honda, January 2002. http://www.bobhowardhonda.com.

[8] H. Borko and M. Bernick. Automatic document classification. *Journal of the ACM*, 10(2):151–162, April 1963.

[9] M.A. Bunge. *Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World*. Reidel, Boston, 1977.

[10] M.A. Bunge. *Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems*. Reidel, Boston, 1979.

[11] A. Cali, D. Calvanese, G. De Giacomo, and M. Lenzerini. On the expressive power of data integration systems. In *Proceedings of 21st International Conference on Conceptual Modeling (ER2002)*, pages 338–350, Tampere, Finland, October 7–11 2002.

[12] D. Calvanese, D. Lembo, and M. Lenzerini. Survey on methods for query rewriting and query answering using views. Technical report, University of Rome, Roma, Italy, April 2001.

[13] S. Castano and V. De Antonellis. ARTEMIS: Analysis and reconciliation tool environment for multiple information sources. In *Proceedings of the Convegno Nazionale Sistemi di Basi di Dati Evolute (SEBD'99)*, pages 341–356, Como, Italy, June 23–25 1999.

[14] S. Castano and V. De Antonellis. A schema analysis and reconciliation tool environment for heterogeneous databases. In *Proceedings of the International Database Engineering and Applications Symposium (IDEAS-99)*, pages 53–62, Montreal, Canada, August 2–4 1999.

[15] S. Castano, V. De Antonellis, M.G. Fugini, and B Pernici. Conceptual schema analysis: Techniques and applications. *ACM Transactions on Database Systems*, 23(3):286–333, September 1998.

[16] S. Chakrabarti. Recent results in automatic Web resource discovery. *ACM Computing Surveys*, 31(4es), December 1999. Article No. 17.

[17] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of the 10th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, October 7–18 1994.

[18] C. Clifton, E. Hausman, and A. Rosenthal. Experience with a combined approach to attribute-matching across heterogeneous databases. In *Proceedings of the 7th IFIP Conference on Database Semantics (DS-7)*, pages 428–, Leysin, Switzerland, October 7–10 1997.

[19] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD'01)*, pages 509–520, Santa Barbara, California, May 21–24 2001.

[20] D.W. Embley. *Object Database Development: Concepts and Principles*. Addison-Wesley, Reading, Massachusetts, 1998.

[21] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.

[22] D.W. Embley, N. Fuhr, C.-P. Klas, and T. Roelleke. Ontology suitability for uncertain extraction of information from multi-record web documents. In *Proceedings of the Workshop on Agenten, Datenbanken und Information Retrieval (ADI'99)*, Rostock-Warnemuende, Germany, September 30–October 1 1999.

[23] D.W. Embley, D. Jackman, and L. Xu. Multifaceted exploitation of metadata for attribute match discovery in information integration. In *Proceedings of the International Workshop on Information Integration on the Web (WIIW'01)*, pages 110–117, Rio de Janeiro, Brazil, April 9–11 2001.

[24] D.W. Embley, Y.S. Jiang, and Y.-K. Ng. Record-boundary discovery in Web documents. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 467–478, Philadelphia, Pennsylvania, May 31–June 3 1999.

[25] D.W. Embley, C. Tao, and S.W. Liddle. Automating the extraction of data from HTML tables with unknown structure. Submitted for publication.

[26] D.W. Embley, C. Tao, and S.W. Liddle. Automatically extracting ontologically specified data from HTML tables with unknown structure. In *Proceedings of the 21st International Conference on Conceptual Modeling (ER2002)*, pages 322–337, Tampere, Finland, October 7–11 2002.

[27] D.W. Embley and L. Xu. Record location and reconfiguration in unstructured multiple-record web documents. In *Proceedings of the Third International Workshop on the Web and Databases (WebDB2000)*, pages 123–128, Dallas, Texas, May 18–19 2000.

[28] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Massachussets, 1998.

[29] J. Fowler, B. Perry, M. Nodine, and B. Bargmeyer. Agent-based semantic interoperability in InfoSleuth. *SIGMOD Record*, 28(1):60–67, March 1999.

[30] W.B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures & Algorithms*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[31] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99)*, pages 67–73, Orlando, Florida, July 18–22 1999.

[32] M.R. Genesereth, A.M. Keller, and O.M. Duschka. Infomaster: An information integration system. In *Proceedings of 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD'97)*, pages 539–542, Tucson, Arizona, May 13–15 1997.

[33] M. Goodman. Prism: A case-based telex classifier. In *Proceedings of the 2nd Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-90)*, pages 25–37, Washington, DC, May 1–3 1990.

[34] A.Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, December 2001.

[35] J. Hammer. The information integration wizard (iwiz) project report work in progress. Technical Report TR99-019, University of Florida, Gainesville, Florida, 1999.

[36] P.J. Hayes and S.P. Weinstein. Construe-TIS: A system for content-based indexing of a database of news stores. In *Proceedings of the 2nd Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-90)*, pages 49–64, Washington, DC, May 1–3 1990.

[37] K.A. Hewett. An integrated ontology development environment for data extraction. Master's thesis, Computer Science Department, Brigham Young University, April 2000.

[38] W. Hoyle. Automatic indexing and generation of classification systems by algorithms. *Information Storage and Retrieval*, 9(4):233–242, April 1973.

[39] J. Hu, R. Kashi, D. Lopresti, G. Nagy, and G. Wilfong. Why table ground-truthing is hard. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 129–133, Seattle, Washington, September 10–13 2001.

[40] R. Hull and G. Zhou. A framework for supporting data integration using the materialized and virtual approaches. In *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'96)*, pages 481–496, Montreal, Canada, June 4–6 1996.

[41] Z. Ives, D. Florescu, M. Friedman, A. Levy, and D. Weld. An adaptive query execution system for data integration. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 299–310, Philadelphia, Pennsylvania, May 31–June 3 1999.

[42] A.Y. Levy, A. Rajaraman, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, pages 251–262, Mumbai (Bombay), India, September 3–6 1996.

[43] W. Li and C. Clifton. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1):49–84, April 2000.

[44] S.W. Liddle, D.W. Embley, and S.N. Woodfield. An active, object-oriented, model-equivalent programming language. In M.P. Papazoglou, S. Spaccapietra, and Z. Tari, editors, *Advances in Object-Oriented Data Modeling*, pages 333–361. MIT Press, Cambridge, Massachusetts, 2000.

[45] S.W. Liddle, S.H. Yau, and D.W. Embley. On the automatic extraction of data from the hidden web. In *Proceedings of the International Workshop on Data Semantics in Web Information Systems (DASWIS-2001)*, pages 106–119, Yokohama, Japan, November 27–30 2001.

[46] J.J. Lu, G. Moerkotte, J. Schue, and V.S. Subrahmanian. Efficient maintenance of materialized mediated views. In *Proceedings of ACM SIGMOD Conference on the Management of Data (SIGMOD'95)*, pages 340–351, San Jose, California, May 22–25 1995.

[47] J. Madhavan, P.A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 49–58, Rome, Italy, September 11–14 2001.

[48] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow.

[49] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, July 2000.

[50] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *Proceedings of the First IFCIS International Conference on*

Cooperative Information Systems (CoopIS'96), pages 14–25, Brussels, Belgium, June 19–21 1996. IEEE Computer Society, New York.

[51] G.A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, November 1995.

[52] R. Miller, L. Haas, and M.A. Hernandez. Schema mapping as query discovery. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, pages 77–88, Cairo, Egypt, September 10–14 2000.

[53] R.J. Miller, M.A. Hernandez, L.M. Haas, L. Yan, C.T. Howard Ho, R. Fagin, and L. Popa. The Clio project: Managing heterogeneity. *ACM SIGMOD Record*, 30(1):78–83, March 2001.

[54] T. Milo and S. Zohar. Using schema matching to simplify heterogeneous data translation. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB-98)*, pages 122–133, New York City, New York, August 24–27 1998.

[55] P. Mitra, G. Wiederhold, and J. Jannink. Semi-automatic integration of knowledge sources. In *Proceedings of the Second International Conference on Information Fusion (FUSION 99)*, Sunnyvale, California, July 6–8 1999.

[56] L. Palopoli, D. Saccà, and D. Ursino. An automatic technique for detecting type conflicts in database schemes. In *Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management (CIKM'98)*, pages 306–313, Bethesda, Maryland, November 3–7 1998.

[57] L. Palopoli, G. Teracina, and D. Ursino. The system DIKE: Towards the semi-automatic synthesis of cooperative information systems and data warehouses. In *Proceedings of ADBIS-DASFAA 2000*, pages 108–117, Prague, Czech Republic, September 5–8 2000.

[58] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.

[59] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, December 2001.

[60] L.F. Rau and P.S. Jacobs. Creating segmented databases form free text for text retrievel. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'91)*, pages 337–346, Chicago, Illinois, October 13–16 1991.

[61] E. Riloff and W. Lehnert. Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems*, 12(3):296–333, 1994.

[62] L. Tanca S. Ceri, G. Gottlob, editor. *Logic Programming and Databases*. Springer-Verlag, Berlin Heidelberg, 1990.

[63] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[64] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous database systems. *ACM Transactions on Database Systems*, 19(2):254–290, June 1994.

[65] Fabrizio Sebastiani. Machine learning in automated text categorisation: a survey. Technical Report IEI-B4-31-1999, Pisa, Italy, 1999.

[66] N. Singh. Unifying heterogeneous information models. *Communications of the ACM*, 41(5), May 1998.

[67] V.C. Storey, D. Dey, H. Ullrich, and S. Sundaresan. An ontology-based expert system for database design. *Data & Knowledge Engineering*, 28(1):31–46, October 1998.

[68] J. Tang. A probabilistic model for binary catogorization of mutliple-record Web documents. Master's thesis, Brigham Young University, Provo, Utah, March 2001.

[69] A. Tomasic, L. Raschid, and P. Valduriez. Scaling access to distributed heterogeneous data sources with DISCO. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):808–823, 1998.

[70] J.D. Ullman. *Principles of Database and Knowledge-Base Systems, Vol. I.* Computer Science Press, New York, 1988.

[71] J.D. Ullman. Information integration using logical views. In F.N. Afrati and P. Kolaitis, editors, *Proceedings of the 6th International Conference on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40, Delphi, Greece, January 8–10 1997.

[72] Y. Wand. A proposal for a formal model of objects. In W. Kim and F.H. Lochovsky, editors, *Object-Oriented Concepts, Databases, and Applications*, pages 537–559. ACM Press, New York, 1989.

[73] Q. Wang. A binary-categorization approach for classifying multiple-record web documents using a probabilistic retrieval model. Master's thesis, Brigham Young University, Provo, Utah, November 2001.

[74] World Wide Wheels classifieds of cars, May 2003. http://wwwheels.com.

[75] S.H. Yau. Automating the extraction of data behind Web forms. Master's thesis, Computer Science Department, Brigham Young University, Provo, Utah, December 2001.

[76] Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom. View maintenance in a warehousing environment. In *Proceedings of ACM SIGMOD Conference on the Management of Data (SIGMOD'95)*, pages 316–327, San Jose, California, May 22–25 1995.