Learning Object Identification Rules for Information Integration

by

Sheila Tejada

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

August 2002

Dedication

In Loving Memory of

Ruth Lewis, John Krieger, Gary Blacksmith and Luis Tejada

Personal Helicon

by Seamus Heaney

As a child, they could not keep me from wells

And old pumps with buckets and windlasses.

I loved the dark drop, the trapped sky, the smells

Of waterweed, fungus and dank moss.

One, in a brickyard, with a rotted board top.

I savoured the rich crash when a bucket

Plummeted down at the end of a rope.

So deep you saw no reflection in it.

A shallow one under a dry stone ditch

Fructified like any aquarium.

When you dragged out long roots from the soft mulch

A white face hovered over the bottom.

Others had echoes, gave back your own call

With a clean new music in it. And one

Was scaresome, for there, out of ferns and tall

Foxgloves, a rat slapped across my reflection.

Now, to pry into roots, to finger slime,

To stare, big-eyed Narcissus, into some spring

Is beneath all adult dignity. I rhyme

To see myself, to set the darkness echoing.

Acknowledgements

I would like to deeply thank my advisors Craig Knoblock and Steve Minton, who have supported me through thick and thin. They have been encouraging, patient and insightful. I want to thank them for for all their advice and feedback, and for being wonderful mentors. I will always be grateful for the freedom they gave me to pursue my research interests. The many lessons that they have taught me will be fondly remembered as they guide me throughout my research career.

Many thanks to the members of the SIMS and Ariadne research groups for all their help over the years, especially Andrew Philpot, Maria Muslea and Jean Oh. I also learned so much from my academic brothers Chun-nan Hsu, Jose Luis Ambite, Naveen Ashish, Ion Muslea, and Greg Barish. The Intelligent Systems Division has become a home-away-from-home, everyone has become like family to me. Intelligent System Division has provided me with so many opportunities. I want to thank Bill Swartout, Yigal Arens and Paul Rosenbloom for their support of my various endeavors. I also need to thank Kevin Knight and Yolanda Gil for being such great mentors to me for all these years.

There has been many people who have lent me their ear to bounce ideas off, in particular, Ric Crabbe, Rebecca Hwa, Mehran Sahami, and Mike Perowitz. Thanks to my indy movie guys - David, Philipp and Alex for keeping me relatively sane, and to my dear friends who have had my back over the years, Jihie, Dongho, Rogelio, Behnam, Joan, Steve, Amali, Leana, Bill, Audrey, Adam and Simon. Gal and Jafar, my two musketeers, I could not have done it without you.

For my family, with the many challenges, obstacles and losses we have had in recent years, finally the taste of success is sweet. Thank you for your love.

The research reported here was supported in part by the United States Air Force under contract number F49620-01-C-0042, in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory under contract/agreement numbers F30602-01-C-0197, F30602-00-1-0504, F30602-98-2-0109, in part by the Air Force Office of Scientific Research under grant number F49620-01-1-0053, in part by a gift from the Microsoft Corporation, and in part by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, cooperative agreement number EEC-9529152. The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

Table Of Contents

List Of Figures

List Of Tables

Abstract

When integrating information from multiple websites, the same data objects can exist in inconsistent text formats across sites, making it difficult to identify matching objects using exact text match. We have developed an object identification system called Active Atlas, which compares the objects' shared attributes in order to identify matching objects. Certain attributes are more important for deciding if a mapping should exist between two objects. Previous methods of object identification have required manual construction of object identification rules or *mapping rules* for determining the mappings between objects, as well as domain-dependent transformations for recognizing format inconsistencies. This manual process is time consuming and error-prone. In our approach, Active Atlas learns to simultaneously tailor both mapping rules and a set of general transformations to a specific application domain, through limited user input. The experimental results demonstrate that we achieve higher accuracy and require less user involvement than previous methods across various application domains.

Chapter 1

INTRODUCTION

Many problems arise when integrating information from multiple information sources on the web [86]. One of these problems is that data objects can exist in inconsistent text formats across several sources. An example application of information integration involves integrating all the reviews of restaurants from the Zagat's Restaurants website with the current restaurant health ratings from the Department of Health's website. To integrate these sources requires comparing the objects from both sources and identifying which restaurants are the same.

Examples of the object identification problem are shown in Figure 1.1. In the first example the restaurant referred to as "Art's Deli" on the Zagat's website may appear as "Art's Delicatessen" on the Health Department's site. Because of this problem, the objects' instances cannot be compared using equality, they must be judged according to text similarity in order to identify if the objects are the same.

When two objects are determined the same, a *mapping* is created between them. There are two types of object identification information needed to determine mappings between the objects. The first type of information is a set of domain-independent *transformations* for judging text similarity. These transformations suggest possible relationships between tokens, e.g. (Prefix "Deli", "Delicatessen") or between phrases, e.g. (Acronym "California Pizza Kitchen", "CPK"). These transformations are used to identify possible mappings between the objects from the two datasets.

The second type of information is the importance of certain attributes or combinations of attributes for deciding on mappings. The examples in Figure 1.1 are each representative of a type of possible mapping found in the restaurant domain. Together, these types of examples demonstrate the importance of certain attributes or combinations of attributes for deciding mappings between objects.

Both sources list a restaurant named "Teresa's," and even though they match exactly on the **Name** attribute, we would not consider them the same restaurant.

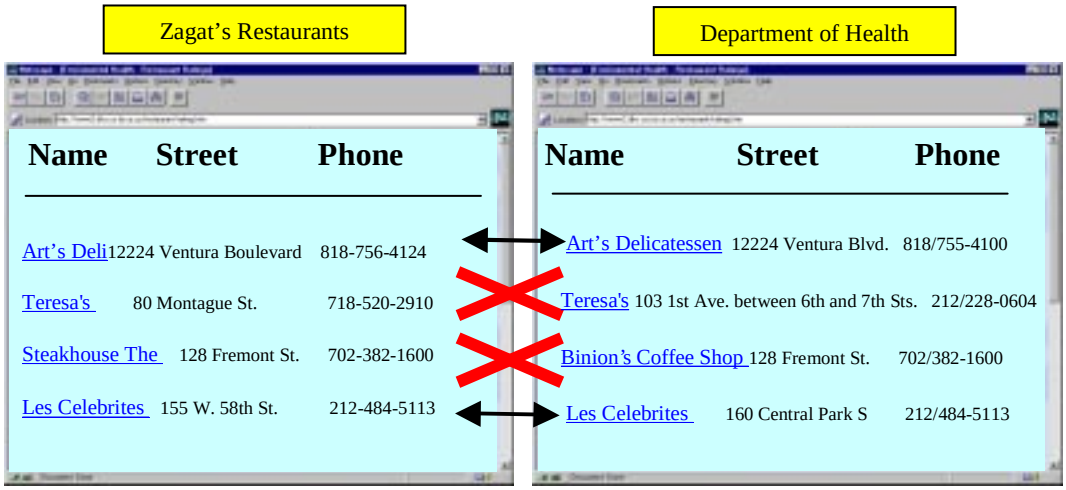

Figure 1.1: Matching Restaurant Objects

These restaurants belong to the same restaurant chain, but they may not share the same health rating. In this restaurant application the **Name** attribute alone does not provide enough information to determine the mappings.

The "Steakhouse The" and "Binion's Coffee Shop" restaurants are located in the same food court of a shopping mall. Although they match on the **Street** and **Phone** attributes, they may not have the same health rating and should not be considered the same restaurant. In the last example, due to error and unreliability of the data values of the **Street** attribute, the restaurant objects match only on the **Name** and **Phone** attributes. Therefore, in order for objects to be correctly mapped together in this application, the objects must match highly on both the **Name** and the **Street** attributes ("Art's Deli") or on both the **Name** and **Phone** attributes ("Les Celebrites"). This type of critical attribute information

is captured in the form of object identification rules (*mapping rules*), which are then used to determine the mappings between the objects.

This thesis presents an object identification system called Active Atlas that starts with a set of general transformations and rules for judging similarity, and then learns to tailor them to a specific application domain in order to determine with high accuracy the set of mappings between the objects of two sources. The main goal of this research is to achieve the highest possible accuracy in object identification with minimal user interaction in order to properly integrate data from multiple information sources. The problem of object identification appears in several research areas as discussed in chapter 5. Active Atlas can be used to handle the object identification problem for a variety of these research areas. For this thesis Active Atlas was applied in conjunction with information mediators, such as SIMS [5] and Ariadne [49], to properly handle the object identification problem for information integration.

1.1 Ariadne Information Mediator

The Ariadne information mediator [50] is a system for extracting and integrating information from sources on the web. Ariadne provides a single interface to multiple information sources for human users or applications programs. Queries to Ariadne are in a uniform language, independent of the distribution of information

over sources, the source query languages, and the location of sources. Ariadne determines which data sources to use and how to efficiently retrieve the data from the sources.

Ariadne has a set of modeling tools that allow the user to rapidly construct and maintain information integration applications for specific information sources, such as the restaurant application shown in Figure 1.2. This application integrates information about restaurants from Zagat's Restaurants with information from the Department of Health. In order to retrieve data objects from these web sources, *wrappers* are created for each source.

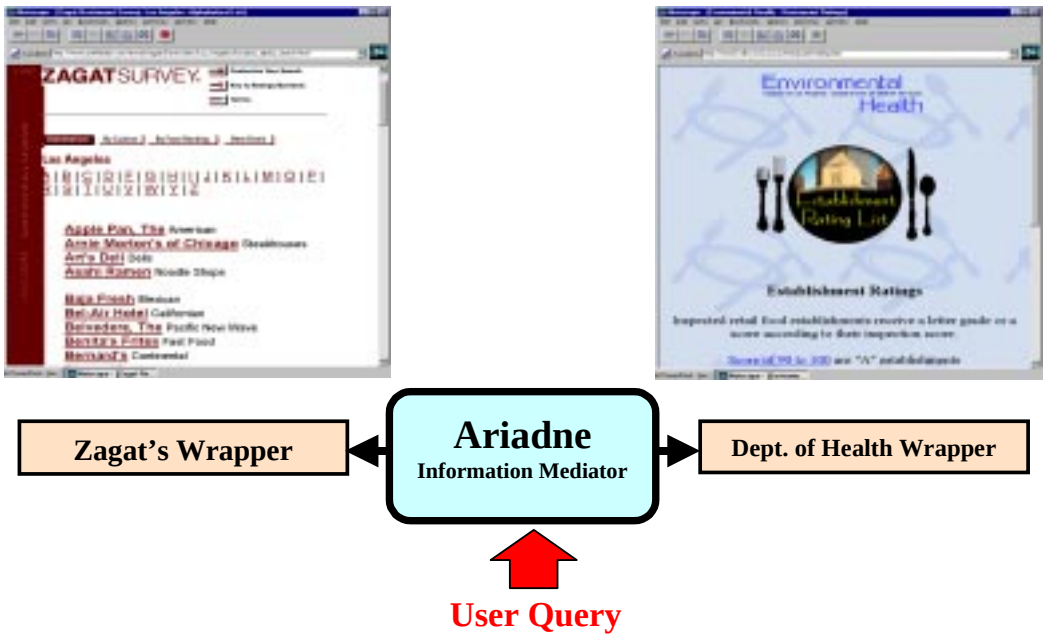

Figure 1.2: Restaurant Application

A wrapper is software that extracts information from a website and provides a database-like interface. In this restaurant application two wrappers are created. Application building tools are provided with the Ariadne system, including a wrapper building tool, which generates a wrapper to properly extract information from a website. When the application has been constructed, the Ariadne information mediator can answer queries from the user by breaking them into individual queries to the appropriate wrapper.

Data objects extracted from websites by these wrappers can be stored in the form of records in a relational table or database (Figure 1.3). These objects (records) represent entities in the real world, like restaurants. Each object has a set of *attributes* (e.g., **Name**, **Street**, **Phone**). A specific restaurant object, for example, may have the following set of values for its attributes: the value "Art's Deli" for the **Name** attribute, the value "12224 Ventura Boulevard" for the **Street** attribute, and the value "818-756-4124" for the **Phone** attribute. As shown in Figure 1.3 the attribute values of objects can have different text formats and values across websites or information sources.

To allow the user to properly query the information mediator about these objects, there is a unifying domain model created for each Ariadne application which provides a single ontology. The domain model (Figure 1.4) is used to describe the contents of the individual sources. Given a query in terms of the concepts (e.g. Restaurant) and attributes (e.g. Name and Phone) described in

Zagat's Restaurant Table

Name	Street	Phone
Art's Deli	12224 Ventura Boulevard	818-756-4124
Teresa's	80 Montague St.	718-520-2910
Steakhouse The	128 Fremont St.	702-382-1600
Les Celebrites	155 W. 58th St.	212-484-5113

Department of Health's Restaurant Table

Name	Street	Phone
Art's Delicatessen	12224 Ventura Blvd.	818/755-4100
Teresa's	103 1st Ave. between 6th and 7th Sts.	212/228-0604
Binion's Coffee Shop	128 Fremont Street	702/382-1600
Les Celebrites	160 Central Park S	212/484-5113

Figure 1.3: Restaurant Data Objects Stored as Records

the model, the system dynamically selects an appropriate set of sources and then generates a plan to efficiently produce the requested data.

Unfortunately, due to the object identification problem described in Figure 1.1, the given domain model (Figure 1.4) does not provide enough information for the mediator to properly integrate the information from the sources. To address this problem, Active Atlas can be applied to determine with high accuracy the mappings between the objects of the sources and add new information sources to the domain model that include the necessary information for integrating the sources (Figure 1.5).

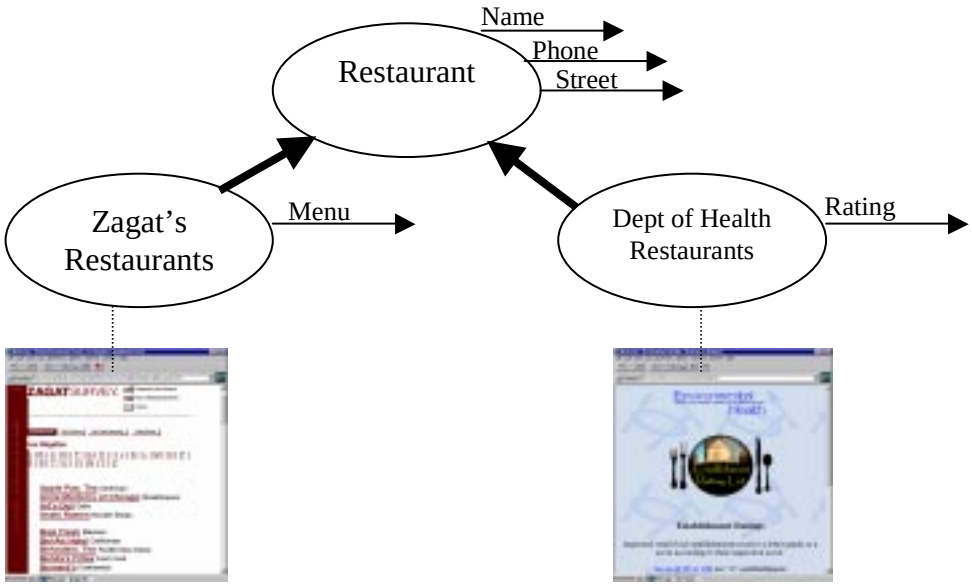

Figure 1.4: Restaurant Domain Model

After Active Atlas determines the total mapping assignment for an application, it builds two types of tables for storing the mapping information in order to properly access and integrate these sources in the future. The mapping information is stored as a global object table and individual source mapping tables. The global object table contains a unique identifier for each object in the application. The global object table represents the union of the objects in the sources, capturing the exact relationship between the sources. In the restaurant application, this table may contain, for examples, the restaurant names as the unique identifiers for the union of the Zagat's and Health Dept's restaurants. Because there is only one entry in the table for each unique restaurant, only one of the duplicate

instances, such as "Art's Deli" and "Art's Delicatessen," can be chosen to represent the restaurant in the global object table. The sources are ranked by user preference, so that the instances from the most preferred source are chosen to be included in the global object table. In this example, the Dept. of Health source has been chosen as the preferred source. Its instances (e.g. "Art's Delicatessen") will be entered into the table over the Zagat's instances (e.g. "Art's Deli") when there are duplicate instances.

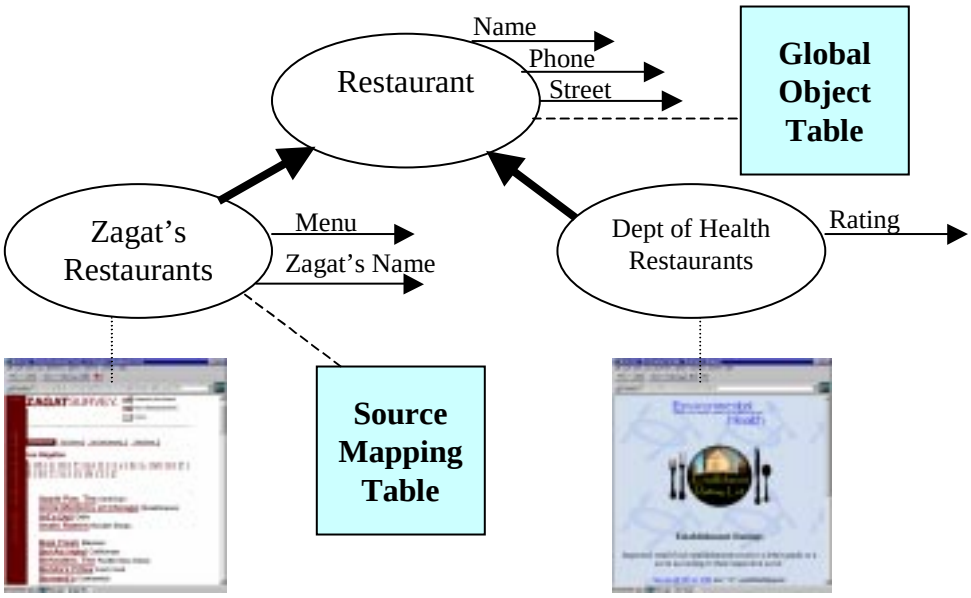

Figure 1.5: Restaurant Domain Model with Mapping Table

Ariadne will now be able to query the sources for restaurant information using the information given in the global object table. Because these queries will refer to restaurants shared by the sources using only preferred source (Dept. of Health) instances, these instances must be translated when querying the other sources

(Zagat's). This type of mapping information is stored as a source mapping table, or as a source mapping function if a compact translation scheme can be found to accurately convert data instances from one source into another.

For the restaurant domain, the source mapping table would relate every object from the Zagat's Restaurant source to its counterpart in the global object table. This mapping table would contain two attributes: Restaurant **Name** and Zagat's **Name**. If the Zagat's restaurant did not have a duplicate in the Dept. of Health source, then Restaurants **Name** and Zagat's **Name** would be the same. Figure 1.5 shows a domain model in which one source (Dept of Health) has a set of unique instances; therefore, the source mapping table captures a one-to-one or one-to-many relationship between the sources, depending on whether the other source (Zagat's) has a set of unique instances or contains duplicates within the source. To represent a many-to-many relationship between the objects, where multiple sources contain duplicate instance, each source will need a source mapping table as shown in Figure 1.6.

Once these mapping constructs, i.e. mapping tables or functions, have been automatically generated, they can be considered new information sources (Figure 1.5). Active Atlas creates these mapping information sources, so that mediators, like Ariadne, can use them to accurately integrate data from inconsistent sources in an intelligent and efficient manner.

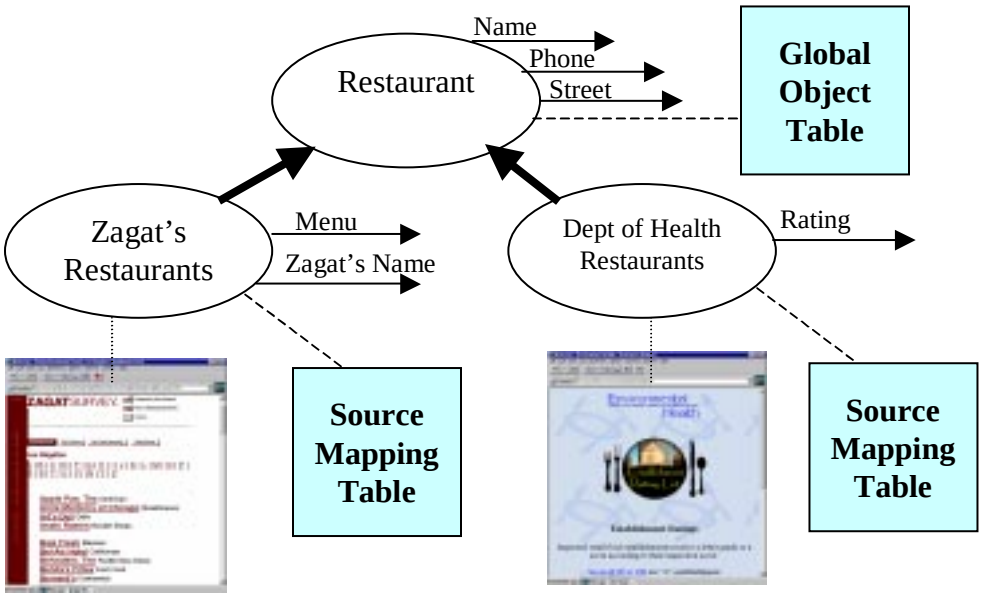

Figure 1.6: Restaurant Domain Model with Multiple Mapping Tables

1.2 Approach

Identifying mappings between objects may be dependent on the application. The information necessary for deciding on a mapping may not be evident by solely evaluating the data itself because there may be knowledge about the task that is not represented in the data. For example, the same Government Census data can be grouped by household or by each individual (Figure 1.7) depending on the task. In the figure below the objects corresponding to **Mrs. Smith** and **Mr. Smith** would not be mapped together if the application is to retrieve information about an individual, such as their personal income, from the Government Census data. But, if the application is to determine the household mailing addresses

in order to mail the new Census 2000 form to each household, then the objects **Mrs. Smith** and **Mr. Smith** would be mapped together, so that the Smith household would only receive a single Census 2000 form.

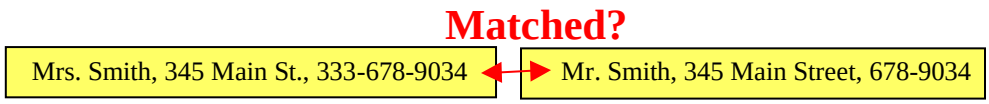

Figure 1.7: Example Census Data

In the restaurant domain (Figure 1.1), because we are retrieving information about health ratings, we are interested in finding the same physical restaurant between the sources. In other words we would not map together restaurants belonging to the same chain, like the "Teresa's" restaurants in Figure 1.1, or the restaurants in the same food court of a shopping mall, like "Steakhouse The" & "Binion's Coffee Shop." Because of these types of examples, a combination of the **Name** attribute and either the **Street** or **Phone** attributes is necessary to determine whether the objects should be mapped together.

But, because the mapping assignment may depend on the specific application, if the application is altered then the mapping assignment may change as well, even though the sets of data are the same. For example, health inspectors planning to visit the restaurants in order to update their health ratings would be interested in obtaining the combined set of restaurant addresses from Zagat's and the Dept. of Health. To efficiently print out a map of directions to each of the

restaurants, all restaurants with the same address would be considered the same, .e.g "Steakhouse The" & "Binion's Coffee Shop." A different set of mapping rules would be needed to properly map the objects because the measurement of the similarity between objects has changed for this application. Since the data itself may not enough to decide the mappings between the sets of objects, the user's knowledge about object similarity is needed to increase the accuracy of the total mapping assignment. We have adopted a general domain-independent approach for incorporating the user's knowledge into the object identification system.

There are two types of knowledge necessary for handling object identification: (1) the importance of the different attributes for deciding a mapping, and (2) the text formatting differences or transformations that may be relevant to the application domain. It is very expensive, in terms of the user's time, to manually encode these types of knowledge for an object identification system. Also, due to errors that can occur in the data, a user may not be able to provide comprehensive information without thoroughly reviewing the data in all sources. The Active Atlas approach to achieving high accuracy object identification is to simultaneously learn to tailor both domain-independent transformations and mapping rules to a specific application domain through limited user input.

1.3 Active Atlas System Overview

Learning the transformations and mapping rules for a specific domain is a circular problem, where the accuracy of the mapping rules depends on the accuracy of the transformations and vice-versa. For Active Atlas this process has two stages as shown in Figure 1.8, applying the transformations to calculate initial similarity scores and then learning the mapping rules and transformations to properly map the objects between the sources. In order to increase mapping accuracy, it is important to accurately weight transformations, because some transformations can be more appropriate for a specific application domain than others. The transformation weights measure the likelihood that if the transformation, like "Equality" or "Acronym," is applied between two objects that those objects will be classified as mapped.

In the first stage the candidate generator is used to propose the set of possible mappings between the two sets of objects by applying the transformations to compare the attribute values. Initially, it is unknown which transformations are more appropriate for a specific domain, so all transformations are treated the same when computing the initial similarity scores for the proposed mappings. The output of the candidate generator is the set of candidate mappings, each with their corresponding set of attribute similarity scores, combined total object

similarity score, and set of applied transformations. The initial similarity scores will be highly inaccurate, but will serve as the basis for the learning to begin.

The second stage is the mapping learner, which learns to tailor both types of object identification information – mapping rules and transformations. Mapping rules determine which attribute or combinations of attributes (**Name, Street, Phone**) are most important for mapping objects by learning the thresholds on the attribute similarity scores computed in the first stage. The purpose of learning the mapping rules is to achieve the highest possible accuracy for object mapping across various application domains. The user's input is necessary for learning these mapping rules. The main idea behind this approach is for the mapping-rule learner to actively choose the most informative candidate mappings, or *training examples*, for the user to classify as mapped or not mapped. The learner constructs high accuracy mapping rules based on these examples, while at the same time limiting the amount of user involvement. Once the rules have been learned, they are applied to the set of candidate mappings to determine the set of mapped objects.

The second type of object identification information learned by the mapping learner is which transformations are appropriate for the domain of the attribute or the application. The general transformations (e.g. Abbreviation, Acronym, and Substring) are domain-independent. The same set of general transformations

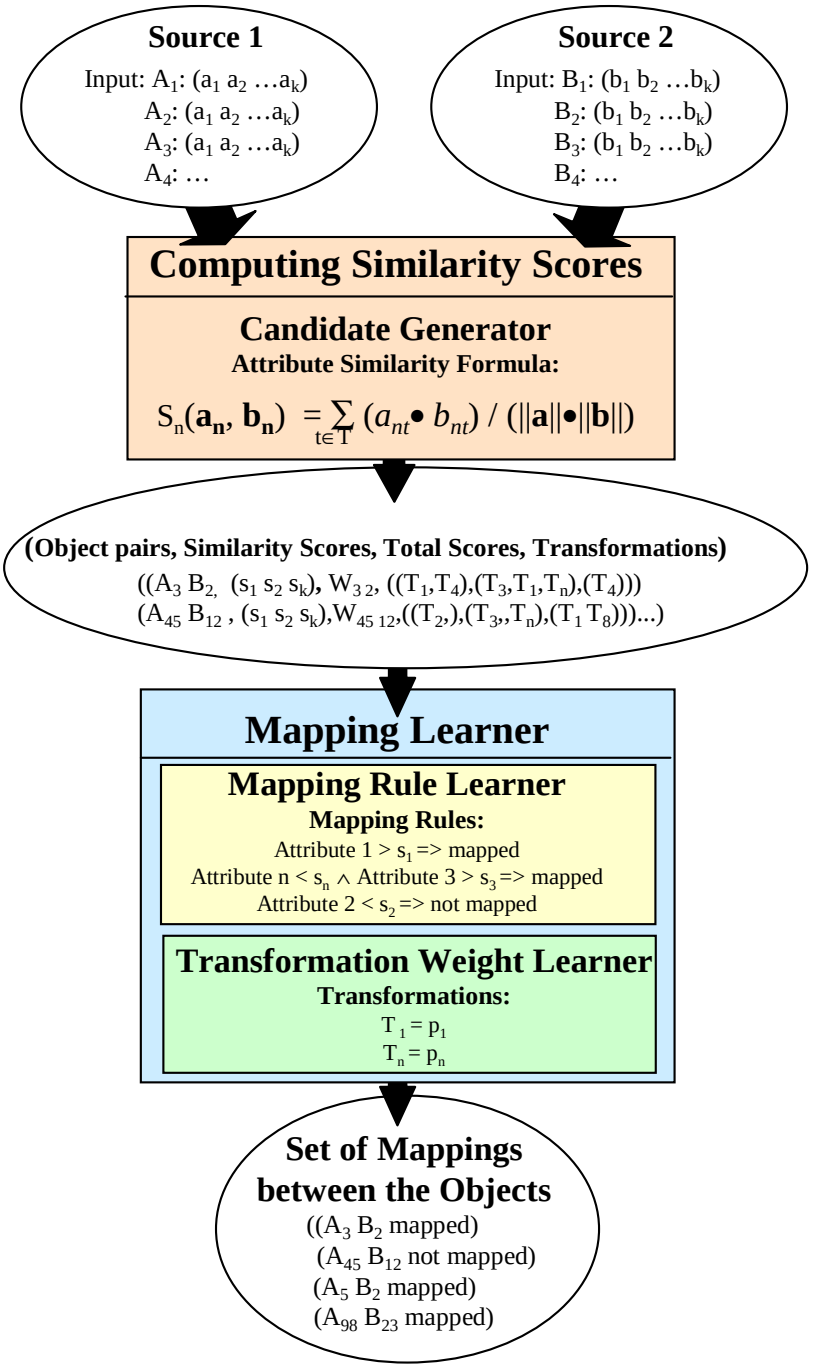

Figure 1.8: General System Architecture

is applied in every application domain and for every attribute. These transformations can suggest possible relationships between tokens, e.g. (Prefix "Deli", "Delicatessen") or between phrases, e.g. (Acronym "California Pizza Kitchen", "CPK"), but these relationships may not accurately represent the true relationship between the tokens. For example, the transformation (Acronym "Crazy Pasta Kitchen", "CPK") may be the actual relationship between the tokens, and not (Acronym "California Pizza Kitchen", "CPK"), though both may be judged with the same similarity score.

Some transformations can also be more appropriate for a specific application domain than others. For example, the transformation Acronym is more appropriate and accurate for the attribute Restaurant Name than for the Phone attribute. This error or bias in token relationships proposed by the transformations is not reflected in the initial attribute similarity scores calculated by the candidate generator. Therefore, in order to increase the accuracy of the similarity scores, as well as the mapping accuracy, it is important to learn how to weight transformations appropriately for the domain.

The transformation weight learner accomplishes this through analyzing both the user-labeled candidate mappings and those mappings classified by the mapping-rule learner to determine how well the token relationships proposed by the transformations predicted a positively classified mapping between two objects. Once

the transformation weights are determined then the learner recalculates the attribute similarity scores of the candidate mappings. The mapping-rule learner can now reclassify the candidate mappings with greater accuracy because the similarity scores more accurately represent the true textual similarity between the objects.

1.4 Contributions

The main contribution of this thesis is combining two forms of learning to produce an efficient and robust high accuracy object identification system. The following are the key contributions of this research:

- General domain independent method for learning domain-specific knowledge for object identification
- Approach to learning mapping rules that achieve high accuracy mapping while minimizing user involvement
- Approach to tailor a general set of transformations to a specific domain application to resolve format inconsistencies for the problem of object identification
- Novel method to combine both forms of learning to create a robust object identification system

1.5 Outline

The remaining chapters provide a detailed description of the Active Atlas object identification system. Chapter 2 explains the process of computing the similarity scores, while chapter 3 describes the mapping learner. Chapter 4 presents the experimental results of the system. Chapter 5 discusses related work, and chapter 6 concludes with a discussion and future work.

Chapter 2

COMPUTING SIMILARITY SCORES

In comparing the objects between the sources, potentially any object in one source may be mapped to any object in the other source. The number of potential mappings can be at most the number of objects in the first source m multiplied by the number of objects in the second source n. Generating the maximal number of mappings is an expensive process and may be unnecessary in order to determine the correct set of mappings. The candidate generator uses the set of domain-independent transformations to judge the similarity between two objects, so that only the most similar *candidate mappings* between the sources are generated. The

main function of the candidate generator is producing a set of quality candidate mappings while at the same time minimizing the number of candidate mappings that will be considered by the mapping learner.

With every mapping the candidate generator also computes information critical to the learning of the object mappings. It keeps a record of the set of transformations that were applied for each mapping, which is essential for learning the transformation weights. It also calculates a set of similarity scores for each mapping, which is necessary for learning the mapping rules. Both types of information are given as input along with each mapping to the mapping learner.

When comparing objects, the alignment of the attributes is determined by the domain model (Figure 1.4). The values for each attribute are compared individually (Figure 2.1 – **Name** with **Name**, **Street** with **Street**, and **Phone** with **Phone**). Comparing the attributes individually is important in reducing the confusion that can arise when comparing the objects as a whole. Words can overlap between the attribute values. For example, some words in the **Name** of the restaurant "The **Boulevard** Cafe" can appear in the **Street** attribute value of another restaurant. Comparing the attributes individually saves computation and also decreases mapping error by reducing the number of candidate mappings considered.

Given the two sets of objects, the candidate generator is responsible for generating the set of candidate mappings by comparing the attribute values of the

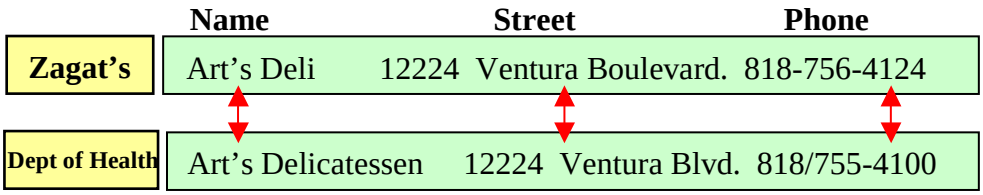

Figure 2.1: Comparing Objects by Attributes

objects. The output of this component is a set of attribute similarity scores for each candidate mapping. A candidate mapping has a computed similarity score for each attribute pair (Figure 2.2). The process for computing these scores is described in detail later in this section.

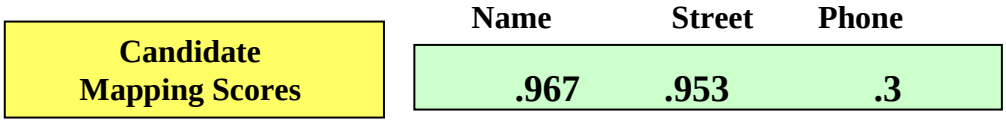

Figure 2.2: Set of Computed Similarity Scores

Figure 2.2 displays example attribute similarity scores for the candidate mapping of the "Art's Deli" and "Art's Delicatessen" objects. The similarity scores for the **Name** and **Street** attribute values are relatively high. This is because the text values are similar and the text formatting differences between them can be resolved. The **Phone** attribute score is low because the two phone numbers only have the same area code in common, and since the dataset contains many restaurants in the same city as "Art's Deli," the area code occurs frequently.

2.1 Candidate Generator Overview

Figure 2.3 depicts the algorithm employed by the candidate generator in order to propose the most similar candidate mappings from the set of all possible mappings. First, two varieties of transformations are applied to the sets of objects to compute the set of transformations needed to relate one object from one source to an object in the other source. A candidate mapping is proposed between those objects that have transformations relating them. This reduces the number of mappings considered by the mapping learner. Next, the attribute similarity scores are calculated for each candidate mapping, given the set of transformations used to generate the candidate mapping. This process repeats for each of the attribute of the objects. In order to compute the total object similarity score that is used to rank the set of candidate mappings, first the *attribute uniqueness weights* are determined. Attribute uniqueness weights are a heuristic measure of the importance of an attribute. After computing the total object similarity scores the candidate generator outputs the set of candidate mappings, each with their corresponding set of attribute similarity scores, combined total object similarity score, and the set of applied transformations.

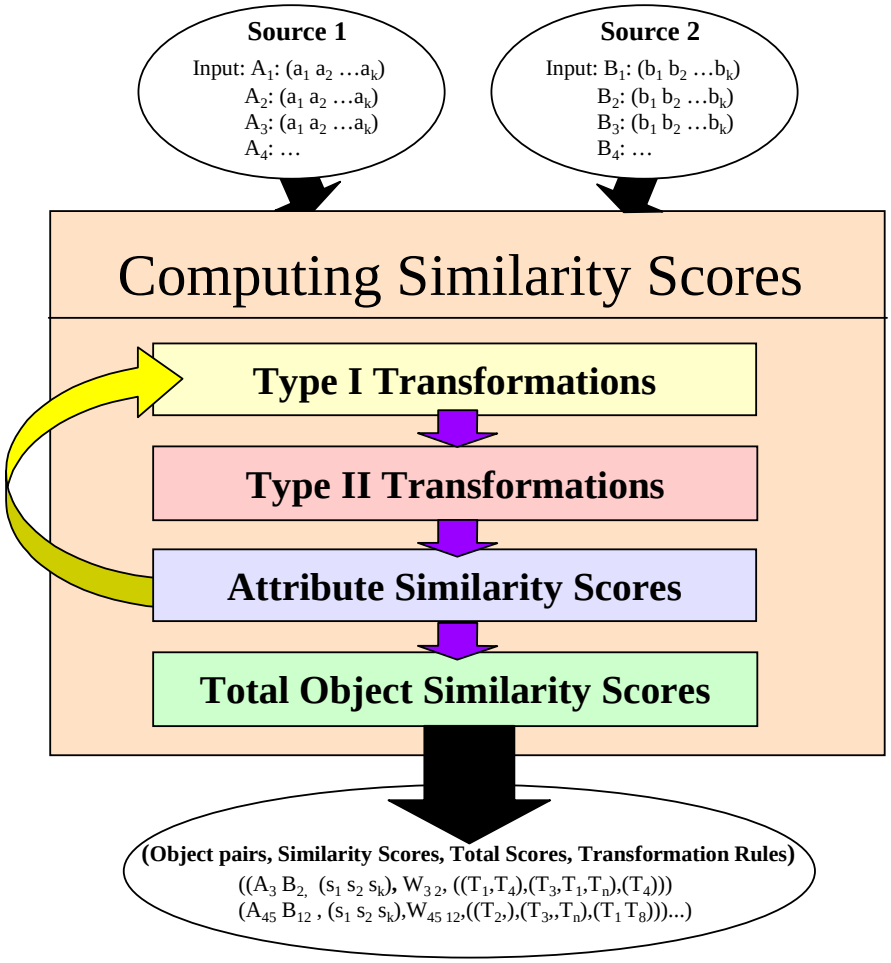

Figure 2.3: Computing Scores Algorithm

2.2 General Transformations

Included in this framework is a set of general domain-independent transformations to resolve the different text formats used by the objects (Figure 2.4). These transformations (e.g. Abbreviation, Acronym, Substring, etc.) are domain-independent and are applied to all of the attribute values in every application

domain and for every attribute. These transformations determine if text transformations exist between words *(tokens)* in the attribute values, e.g., (Prefix - "Deli", "Delicatessen") or between phrases, e.g. (Acronym - "California Pizza Kitchen", "CPK"). If transformations exist between the tokens, then a candidate mapping is proposed between the corresponding objects.

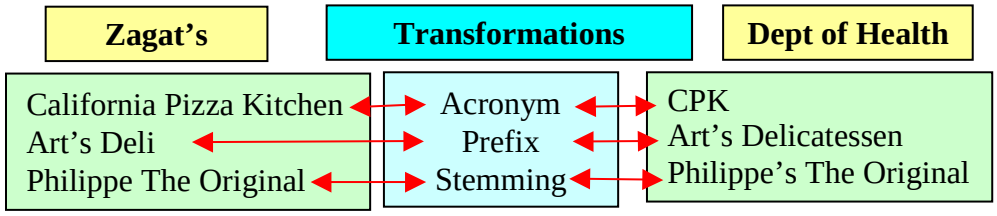

Figure 2.4: Transformations

There are two basic types of the transformations. Unary transformations require only a single token as input in order to compute its transformation. N-ary transformations compare multiple tokens from two objects.

Unary transformations

- **Equality** tests if a token contains the same characters in the same order.
- **Stemming** converts a token into its stem or root.
- **Soundex** converts a token into a Soundex code. Tokens that sound similar have the same code.

- **Abbreviation** looks up a token in a dictionary and replaces the token with corresponding abbreviation (e.g., 3rd or third).

N-ary transformations

- **Initial** computes if one token is equal to the first character of the other.
- **Prefix** computes if one token is equal to a continuous subset of the other starting at the first character.
- **Suffix** computes if one token is equal to a continuous subset of the other starting at the last character.
- **Substring** computes if one token is equal to a continuous subset of the other, but does not include the first or last character.
- **Computed Abbreviation** computes if one token is equal to a subset of the other (e.g., Blvd, Boulevard).
- **Acronym** computes if all characters of one token are initial letters of all tokens from the other object, (e.g., CPK, California Pizza Kitchen).
- **Drop** determines if a token does not match any other token

2.3 Generating Candidate Mappings

The candidate generator uses information retrieval techniques [8] to apply a variety of transformations to resolve text formatting inconsistencies and generate the candidate mappings. An information retrieval engine is used to apply the transformations between sets of attribute values individually, i.e. **Name** with **Name, Street** with **Street** and **Phone** with **Phone** in the restaurant application (Figure 2.1).

Most information retrieval systems, such as the Whirl system [19], apply only the stemming transformation to compare the words of the attribute values. The stemming transformation compares the stem or root of the words to determine similarity. A single transformation Stemming [76] is the only transformation used to calculate similarity between strings; therefore, in Figure 2.4 "CPK" would not match "California Pizza Kitchen." Our approach includes a set of general transformations, so that the system is able to resolve a variety of text formatting differences.

For the candidate generator, the attribute values are considered short documents. These documents are divided into tokens. The tokenization process is to first lowercase all characters and remove punctuation, so that the instance or document "Art's Deli" would produce the following three token list ("art" "s" "deli"). Once the instances have been tokenized, they are then compared using

the transformations. If a transformation exists between the tokens or if the tokens match exactly, then a candidate mapping is generated for the corresponding objects. For the example below, a candidate mapping is generated for the objects "Art's Deli" and "Art's Delicatessen," because there are transformations between their tokens: (Equality – "Art", "Art"), i.e. exact text match, (Equality – "s", "s"), and (Prefix – "Deli", "Delicatessen") (Figure 2.5).

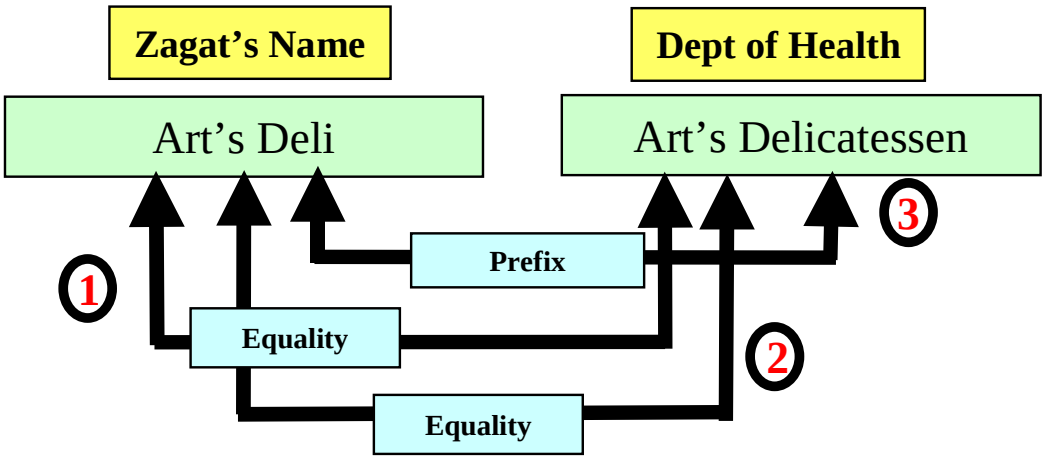

Figure 2.5: Applying Transformations

The candidate generator applies the transformations in a three-step process. The first step is to apply all of the unary transformations to the data. The second step, after applying the unary transformations, is to determine the set of object pairs that are related. And the third step is to apply the n-ary transformations in order to further strengthen the relationship between the two data objects.

2.3.1 Applying Unary Transformations

In step 1, an inverted index or hash-table is created from the tokens of all the instances or documents of one of the attribute datasets, such as the Department of Health's restaurant names. As with a traditional IR engine, the data is first tokenized. The unary transformations are then applied to each token (Equality, Stemming, Soundex, and Abbreviation) and stored in an inverted index. Therefore, each token can have more than one entry in the hash-table. In this index, information about which transformations were applied is stored, as well as the document (object) number.

Table 2.1 shows the restaurant name "Art's Delicatessen" from the Health Dept's document set being tokenized and entered into the index. Each of the four unary transformations are applied to all of the tokens in the order listed previously, increasing the set of tokens to be added to the hash-table. The transformed token set is a unique set of tokens. If a transformation is applied and the transformed token already exists in the set, it is then removed. In Table 2.1 a partial set of the index entries for the Health Dept restaurant names are shown with the tokens for "Art's Delicatessen" in boldface.

Now the other dataset for this attribute (Zagat's restaurant names) is used as a query set against this hash-table. Each object in the Zagat's dataset is tokenized and the transformations are applied to each of the tokens. The object's tokens and the new transformation tokens are both used to query the inverted index

Example:
Document (Object) 5 Restaurant name: "Art's Delicatessen"
Tokens: "Art", "s", "Delicatessen"
Transformed Token Set: "Art", "A630", "s", "S000", "Delicatessen", "D423"

Transformed Tokens	Transformations	Original Tokens	Object Number
"Art"	Equal	"Art"	5
	Stemming	"Arte"	57
"A630"	Soundex	"Art"	5
"s"	Equality	"s"	5,6,9,71,79,97,111
"S000"	Soundex	"s"	5,6,9,71,79,97,111
"Del"	Stemming	"Dell"	57
	Stemming	"Deli"	7,93
	Equality	"Del"	60
"Deli"	Equality	"Deli"	7,93
"D400"	Soundex	"Deli"	7,93
	Soundex	"Dell"	57
"Delicatessen"	Equality	"Delicatessen"	5
"D423"	Soundex	"Delicatessen"	5
"Dell"	Equality	"Dell"	57

Table 2.1: Health Dept's Restaurant Name Inverted Index

of Health Dept Restaurant names. The hash-table entries returned contain the object numbers of the related objects from the other dataset (Department of Health). Table 2.2 lists the retrieved entries from the Health Dept index for each of the tokens for the Zagat's query document "Art's Deli." With this information the set of related Health Dept documents for this query can now be computed. This process is completed for every attribute (**Name**, **Street**, and **Phone**).

At this stage there is a common method used in the IR community which can be employed by the candidate generator to further reduce the number of

Example :

Query: "Art's Deli" (Zagat's)

Tokens: "Art", "s", "Deli"

Transformed Token Set: "Art", "A630", "s", "S000", "Deli", "Del", "D400"

Zagats	Transformations	Health Dept	Document
"Art"	Equality	"Art"	5
	Soundex	"Art"	5
	Stemming	"Arte"	57
"s"	Equality	"s"	5,6,9,71,79,97,111
	Soundex	"s"	5,6,9,71,79,97,111
"Deli"	Equality	"Deli"	7,93,
	Soundex	"Deli"	7,93
	Stemming	"Dell"	57
	Stemming	"del"	60

Table 2.2: Retrieved Health Dept Index Entries for "Art's Deli"

candidate mappings. This method is to have a *stoplist*, where very frequent specific transformations such as (Equality – "Blvd", "Blvd"), or in the case of the transformed token "s" in Table 2.2, does not produce candidate mappings, but are only used to calculate the attribute similarity scores. This method prunes away any candidate mapping which contain only transformations on the stoplist.

A stoplist can be given as input to the system already containing transformations that would frequently occur and yet would not propose quality mappings. Stoplists can also be generated automatically during the creation of the inverted index. A frequency threshold is given, and any transformation whose frequency exceeds the threshold is then added to the stoplist. Frequency is calculated by counting the number of documents that have applied the transformation as shown

in Table 2.2. The candidate generator handles a combination of both techniques, where a stoplist can be given and then automatically augmented when the inverted index is created.

2.3.2 Computing Set of Related Objects

At the end of the first step, the set of related objects and the transformations used to relate them are known for each attribute. The second step is to determine the total set of related objects by combining the sets computed for each of the attributes. In Table 2.3 are shown the sets of related objects computed for each attribute.

Example :

Name:"Art's Deli"
Street:"12224 Ventura Boulevard"
Phone:"818-756-4124"

Name Documents	Street Documents	Phone Documents
5,7,57,60,93	5,7,12,200,	7,49,77,84,92

Table 2.3: Related Health Dept Documents by Attribute

The combined set of related objects for "Art's Deli" is (5, 7, 12, 49, 57, 60, 77, 84, 92, 93, 200). A candidate mapping is generated between the object of "Art's Deli" and each object in the set. Table 2.4 shows a list of the restaurant names for each of the candidate mappings.

Query: "Art's Deli" (Zagat's)

Documents	Restaurant Name
5	"Art's Delicatessen"
7	"Carnegie Deli"
12	"Cava"
49	"California Pizza Kitchen"
57	"Trattoria Dell'Arte"
60	"Ca'del Sol"
77	"Posto"
84	"Border Grill"
92	"Campanile"
93	"Broadway Deli"
200	"Hard Rock Cafe"

Table 2.4: Restaurant Names of Candidate Mappings

Table 2.5 shows the set of candidate mappings, each listed with the best set of unary transformations that relate the attribute values (documents). For some pairs of tokens there are more than one way to relate them, as shown in Table 2.2. For example, both the transformations Equality and Soundex relate the tokens "Art" and "Art." The higher ranked transformation is chosen always as the preferred method for relating tokens. The transformations are ranked by the order in which they are applied (Equality, Stemming, Soundex, Abbreviation), so the transformation Equality will be chosen to relate the tokens "Art" and "Art."

In Table 2.5 some of the candidate mappings do not have any transformations relating them to any of the tokens of the query "Art's Deli" for the Restaurant Name Attribute. Also, for some of the mappings with transformations there

Query: "Art's Deli" (Zagat's)

Document	Restaurant Name	Hypothesis transformation Set
5	"Art's Delicatessen"	("Art" Equality "Art") ("s" Equality "s")
7	"Carnegie Deli"	("Deli" Equality "Deli")
12	"Cava"	
49	"California Pizza Kitchen"	
57	"Trattoria Dell'Arte"	("Art" Stemming "Arte") ("Deli" Stemming "Dell")
60	"Ca'del Sol "	("Deli" Stemming "del")
77	"Posto"	
84	"Border Grill"	
92	"Campanile"	
93	"Broadway Deli"	("Deli" Equality "Deli")
200	"Hard Rock Cafe"	

Table 2.5: Candidate Mappings with Unary Transformations

are tokens that do not relate to the query. The next step is to apply n-ary transformations in order to relate more tokens and increase the measurement of textual similarity. Computing the sets of transformation is important for calculating the attribute similarity scores, as well as, for transformation weight learning by the mapping learner.

2.3.3 Applying N-ary Transformations

After the set of candidate mappings have been determined with the unary transformations, then the more computationally expensive transformations (n-ary) are applied to the remaining unmatched tokens of the attribute values for the related object pairs in order to improve the match. When this step is finished, it will

output the set of related object pairs with their corresponding set of transformations used to related them, along with the token frequency information needed for computing the similarity scores.

Table 2.6 shows a subset of the candidate mappings for the query object "Art's Deli." The n-ary transformations are shown in boldface. The candidate mappings not depicted in the table are the ones which involved dropping all of the tokens from both documents because there were no other ways of relating the tokens for the **Name** attribute. In these cases the tokens relate for other attributes, such as the **Street** or **Phone** attributes, which is the reason the candidate mapping was proposed.

This process of generating a set of candidate mappings by applying the transformations is performed for each object of the Zagat's dataset across all of the attributes. The transformations sets computed for each of the attributes of the candidate mapping for the objects "Art's Deli" and "Art's Delicatessen" is depicted in Figure 2.6. Once these sets have been computed for every candidate mapping, then the attribute similarity scores can be calculated. Also, keeping record of the set of transformations that were applied for each mapping will later allow the transformation weight learner to measure the likelihood that if the transformation is applied, the objects will be classified as mapped.

Query: "Art's Deli" (Zagat's)

Document	Restaurant Name	Hypothesis Rule Set
5	"Art's Delicatessen"	("Art" Equality "Art") ("s" Equality "s") **("Deli" Prefix "Delicatessen")**
7	"Carnegie Deli"	("Deli" Equality "Deli") **(Drop "Carnegie")** **(Drop "Art")** **(Drop "s")**
57	"Trattoria Dell'Arte"	("Art" Stemming "Arte") ("Deli" Stemming "Dell") **(Drop "Trattoria")** **(Drop "s")**
60	"Ca'del Sol"	("Deli" Stemming "del") **("s" Initial "Sol")** **(Drop "Art")**
93	"Broadway Deli"	("Deli" Equality "Deli") **(Drop "Broadway")** **(Drop "Art")** **(Drop "s")**

Table 2.6: Candidate Mappings with N-ary Transformations

2.4 Computing Attribute Similarity Scores

This section discusses in detail how the computed sets of transformations are used to calculate the attribute similarity scores. Figure 2.7 shows how attribute values, ($\mathbf{Z_{name}}$, $\mathbf{Z_{street}}$, $\mathbf{Z_{phone}}$) and ($\mathbf{D_{name}}$, $\mathbf{D_{street}}$, $\mathbf{D_{phone}}$), are compared in order to generate the set of attribute similarity scores ($\mathbf{S_{name}}$, $\mathbf{S_{street}}$,$\mathbf{S_{phone}}$). In Figure 2.7, $\mathbf{Z_{name}}$ and $\mathbf{D_{name}}$ represent **Name** attribute values, e.g. "Art's Deli" and "Art's Delicatessen." These values are compared

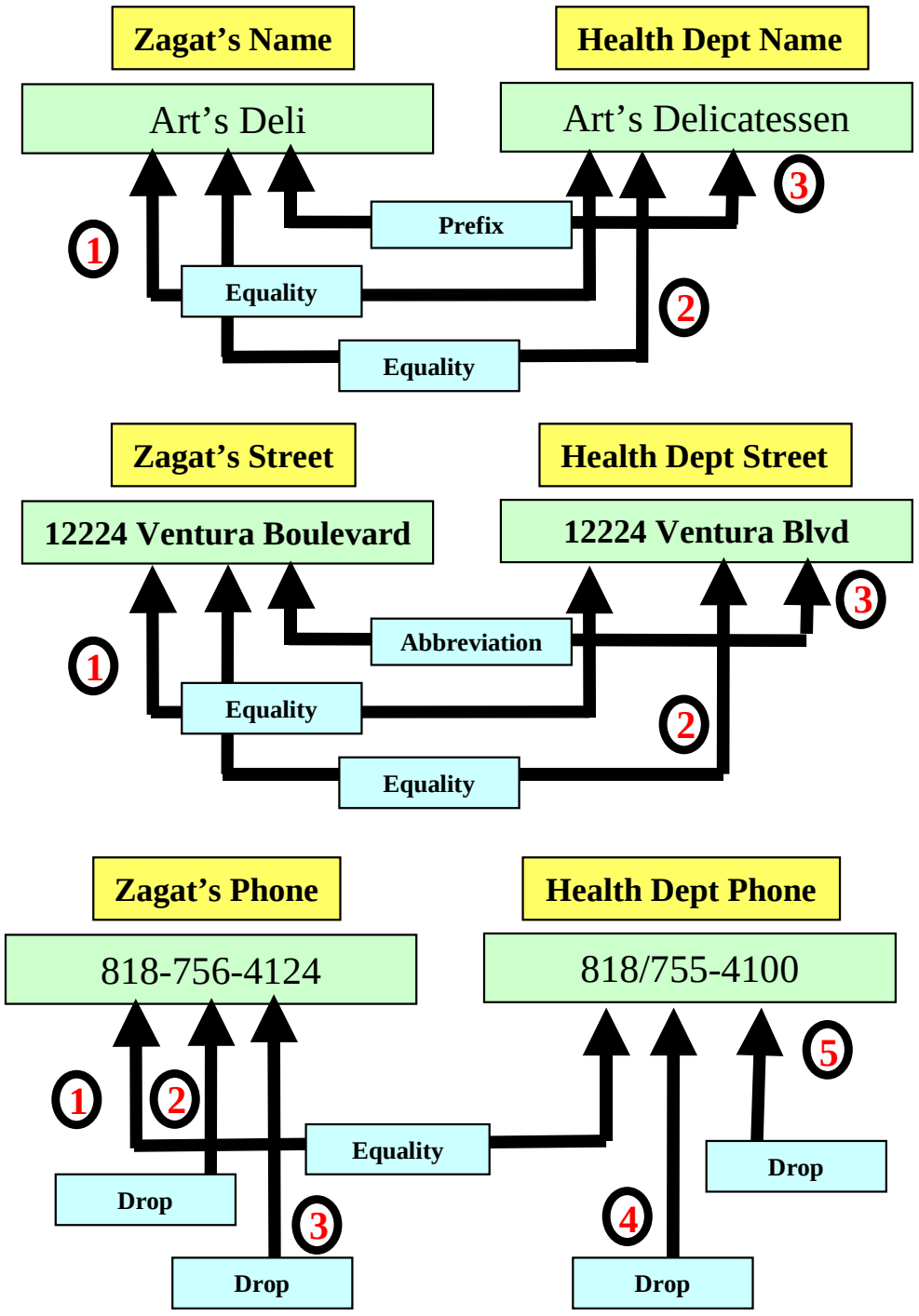

Figure 2.6: Attribute Transformation Sets

using general transformations and then the set of computed transformation are used to calculate the similarity score $\mathbf{S_{name}}$.

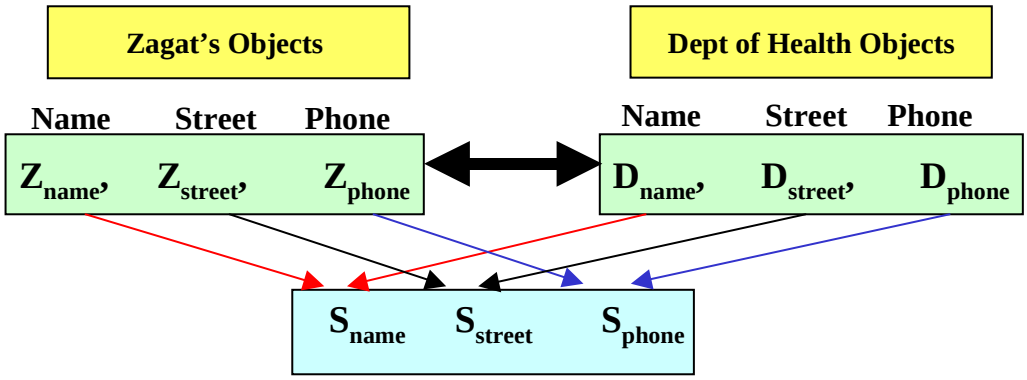

Figure 2.7: Computing Similarity Scores

The candidate generator employs the cosine measure commonly used in information retrieval engines with the TFIDF (Term Frequency x Inverse Document Frequency) weighting scheme [29] to calculate the similarity of each of the objects. Because the attribute values of the object are very short, the within-document term frequency weighting is binary. The within-document frequency is 1 if the term exists in the document and 0 otherwise. The similarity score for a pair of attribute values is computed using the following attribute similarity formula:

$$Similarity(A, B) = \frac{\sum_{i=1}^{t}(w_{ia} \bullet w_{ib})}{\sqrt{\sum_{i=1}^{t} w_{ia}^2 \bullet \sum_{i=1}^{t} w_{ib}^2}}$$

- $w_{ia} = (0.5 + 0.5 freq_{ia})$ x IDF
- $w_{ib} = freq_{ib}$ x IDF_i
- $freq_{ia}$ = frequency for token i of attribute value **a**
- IDF_i= IDF (Inverse Document Frequency) of token i in the entire collection
- $freq_{ib}$ = frequency of token i in attribute value **b**

In this formula **a** and **b** represent the two documents (attribute values) being compared and **t** represents the total number of transformations in the document collection. The terms w_{ia} andw_{ib} correspond to the weights computed by the TFIDF weighting function. This formula measures the similarity by computing the distance between two attribute values.

2.5 Calculating Total Object Similarity Scores

When the candidate generator is finished, it outputs all of the candidate mappings it has generated along with each of their corresponding set of attribute similarity scores. Example sets of attribute similarity scores from the candidate generator are shown in Table 2.7. For each candidate mapping, the total object similarity score is calculated as a weighted sum of the attribute similarity scores.

Each attribute has *a uniqueness weight* that is a heuristic measure of the importance of that attribute. This is to reflect the idea that we are more likely

Restaurant Mappings	(Name, Street, Phone, Total)
(“Les Celebrites”,“Les Celebrites”)	(1.0, .43, 1.0,2.9)
(“Art’s Deli”,“Art’s Delicatessen”)	(.97, .95, .3, 2.8)
(“Spago”,“Spago (Los Angeles)”)	(.83, .2, 1.0, 2.7)
(“Teresa’s”,“Teresa’s”)	(1.0, .12 , .3, 1.7)
(“Jan’s Restaurant”,“Joe’s Restaurant”)	(.17, .3, .74, 1.2)

Table 2.7: Example Output of the Candidate Generator

to believe mappings between unique attributes because the values are rarer. The uniqueness weight of an attribute is measured by the total number of unique attribute values contained in the attribute set divided by the total number of values for that attribute set. There are two uniqueness weights for each attribute similarity score, because we are comparing pairs of attribute values – one from each of the two sources being integrated. To calculate the total object similarity score, each attribute similarity score is multiplied by its associated uniqueness weights, and then summed together, as shown in the following formula:

$$TotalObjectSimilarity(S, W) = \sum_{i=1}^{n} s_i w_{ia} w_{ib}$$

- S = set of attribute similarity scores, size n
- n = number of object attributes
- W = set of uniqueness weight pairs

Once the total object scores are computed, the set of candidate mappings can be reduced further by simply setting a limit on the maximum number of mappings per object. This reduced set of the candidate mapping information is then ranked according to the object similarity score and given as input to the mapping learner (Figure 2.7).

Chapter 3

LEARNING OBJECT MAPPINGS

The goal of learning object mappings is to identify with high accuracy those pairs of objects that are the same from the set of candidate mappings proposed by the candidate generator. The candidate generator provides the necessary information for learning object mappings. Given with each candidate mapping are the set of computed attribute similarity scores and the set of transformations applied between the objects. From this information both types of object identification information, mapping rules and transformations, are tailored to a specific application domain.

Mapping rules are created to classify the candidate mappings as mapped or not mapped based on the attribute similarity scores. The mapping rules determine which attributes, or combination of attributes, are needed to accurately classify the candidate mappings. At the time the attribute similarity scores are computed, it is not known which transformations are appropriate for the application domain, and therefore the attribute similarity scores may not accurately represent the true similarity between all of the objects.

Accurate transformation weights for the specific domain must be known in order to accurately compute new attribute similarity scores. Once the set of candidate mappings have been classified as mapped or not mapped, then the transformation weights can be determined by combining information from both user-labeled mappings and those labeled by the mapping-rule learner. By computing the number of times the transformation was applied between objects that are mapped, divided by the total number of times the transformation was applied, the transformation weight learner calculates weights for the transformations that are then used for recomputing the attribute similarity scores.

This is a circular problem, where the accuracy of the mapping rules depends on the accuracy of the transformations and vice-versa. To address this problem Active Atlas employs an active learning technique that iteratively refines both the mapping rules and transformation weights in order to classify the set of candidate mappings. The set of candidate mappings, produced by the candidate generator,

serves as the basis for the learning to begin. Given these initial attribute similarity scores, mapping rules can be created to classify the mappings. Using the classification information, new transformation weights can be calculated, allowing for new attribute similarity scores and mapping rules. At every iteration of the algorithm, the mapping rules and transformations can more accurately capture the relationships between the objects.

As shown in (Figure 3.1) the mapping learner combines two types of learning, the mapping-rule learning and the transformation weight learning, into one active learning system. The mapping learner incrementally learns to classify the mappings between objects by offering the user one example to label at a time, and from those examples learning both the mapping rules and transformation weights. The criteria for choosing the next example for the user to label is determined by input from both the mapping-rule learner and the transformation weight leaner.

3.1 Mapping-Rule Learner

The mapping-rule learner determines which attribute, or combinations of attributes (**Name**, **Street, Phone**), are most important for mapping objects. The purpose of learning the mapping rules is to achieve the highest possible accuracy for object mapping across various application domains. In this approach,

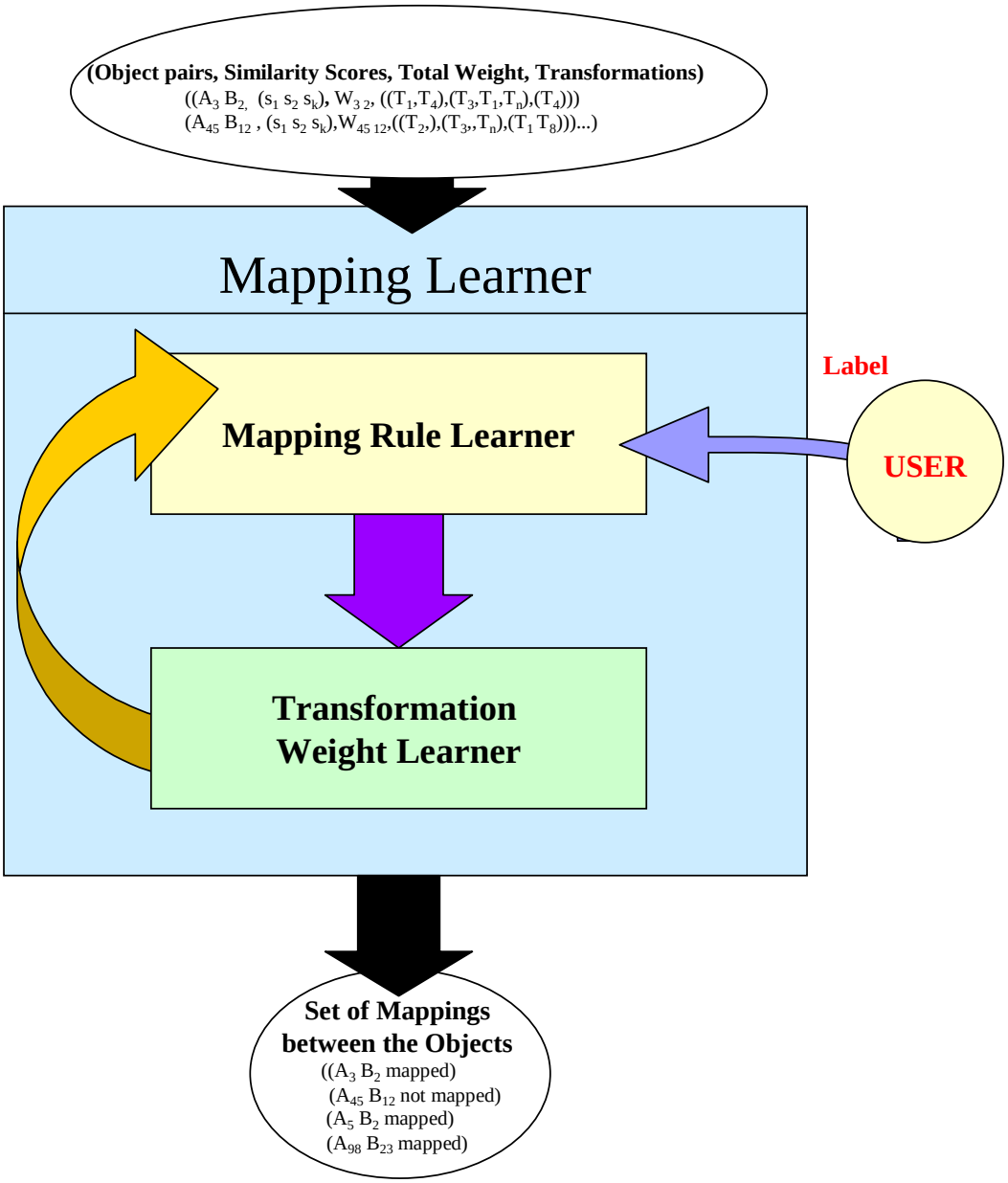

Figure 3.1: Mapping Learner

the system actively chooses the most informative candidate mappings (*training examples*) for the user to classify as mapped or not mapped in order to minimize

the number of user-labeled examples required for learning high accuracy mapping rules.

The mapping-rule learner consists of a committee of decision tree learners. Each decision tree learner creates its own set of mapping rules from training examples labeled by the user. The mapping rules classify an example as mapped or not mapped. These classifications are used by the transformation weight learner for increasing the accuracy of the transformation weights, and are also needed for deciding which training examples should be labeled.

3.1.1 Decision Tree Learning

Mapping rules contain information about which combination of attributes are important for determining the mapping between two objects, as well as, the thresholds on the similarity scores for each attribute. Several mapping rules may be necessary to properly classify the objects for a specific domain application. Examples of mapping rules for the restaurant domain are:

- **Rule 1: Name** $> .859$ and **Street** $> .912 \Longrightarrow$ mapped
- **Rule 2: Name** $> .859$ and **Phone** $> .95 \Longrightarrow$ mapped

These rules are obtained through *decision tree learning* [78]. Decision tree learning is an inductive learning technique, where a learner constructs a decision tree (Figure 3.2) to correctly classify given positive and negative labeled examples.

Decision trees classify an example by starting at the top node and traversing the tree down to a leaf, which is the classification for the given example. Each node of the tree contains a test to perform on an attribute, and each branch is labeled with a possible outcome of the test.

To classify for the candidate mapping of the objects "Art's Deli" and "Art's Delicatessen," which has the attribute similarity scores (.967 .953 .3), we can use the decision tree shown in Figure 3.2. First, the **Name** attribute is tested. Its result is positive, so the **Street** attribute is tested next. Its result is also positive, and we follow the positive branch to reach a leaf node with a *mapped* classification; therefore, this example is classified as mapped.

Decision trees are created by determining the most useful attributes for classifying the examples. A metric called *information gain* measures how well an attribute divides the given set of training examples by their classification (mapped or not mapped). Creating the decision tree is an iterative process, where the attribute with the greatest information gain is chosen at each level. Once a decision tree is created, it can be converted into mapping rules, like those described above.

3.1.2 Active Learning

To efficiently learn the mapping rules for a particular task or domain, we are currently applying a supervised learning technique, which uses a combination

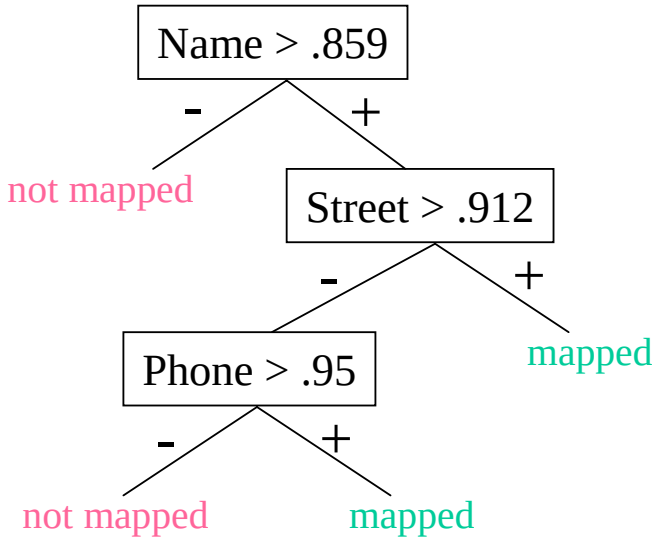

Figure 3.2: Example Decision Tree for Restaurant Domain

of several decision tree learners, based on an algorithm called *query by bagging* [1](Figure 3.3). This technique generates a committee of decision tree learners that vote on the most informative example or candidate mapping for the user to classify next. A single decision tree learner on its own can learn the necessary mapping rules to properly classify the data with high accuracy, but may require a large number of user-labeled examples, as shown in the Experimental Results chapter.

Query by bagging is considered an active learning technique [4] because the system actively chooses examples for the user to label. More specifically, it is a *selective sampling* method [30] because examples are chosen from a fixed set

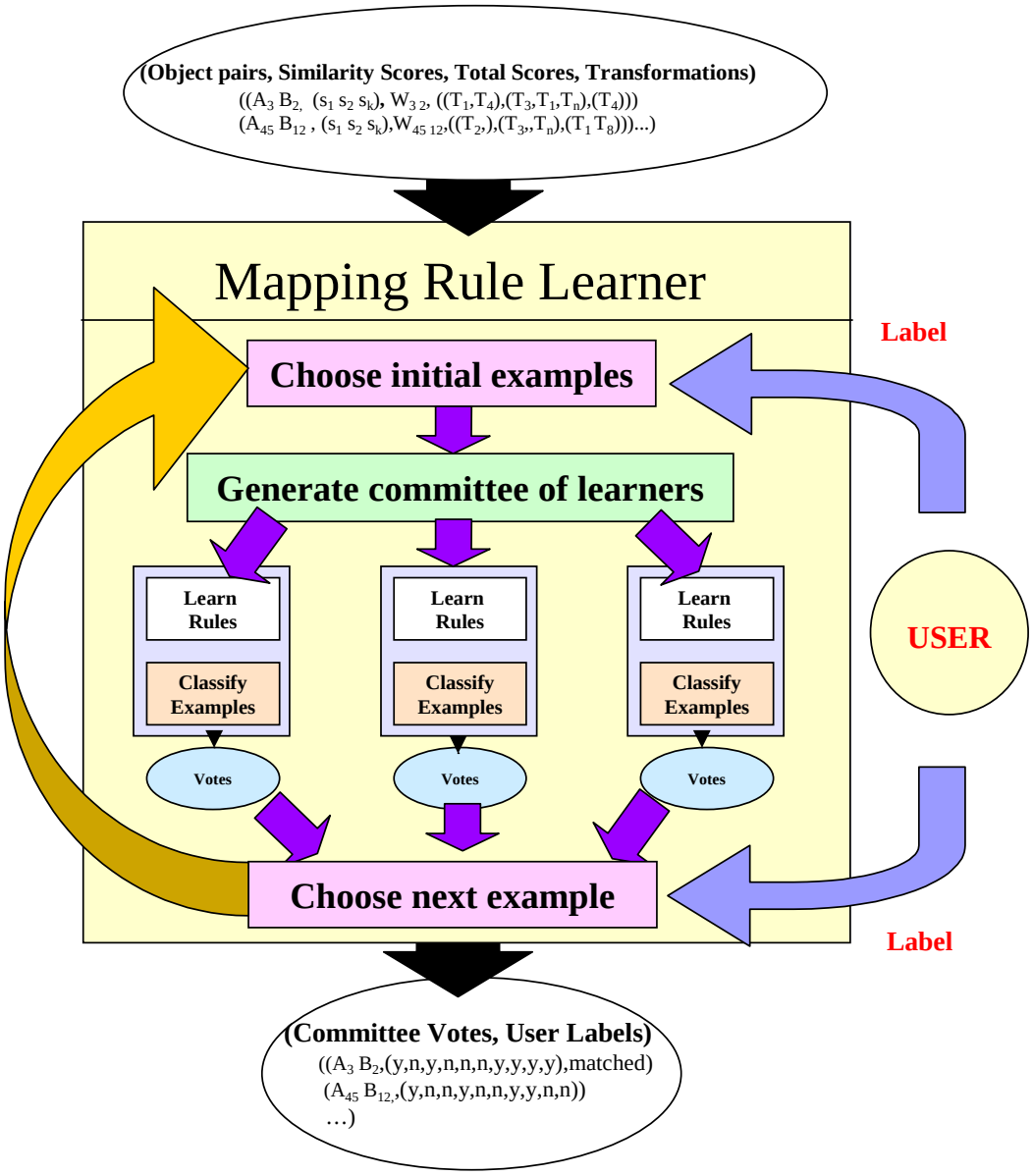

Figure 3.3: Mapping-Rule Learner

of training examples. Query by bagging combines two techniques: *query by committee* [82] and *bagging* [12].

The query by committee approach uses the inputs of several learners to decide the classification of the examples. In general the examples are classified according to how the majority of learners classified the example. The degree to which the learners' classifications disagree is used to determine how informative an example would be if it was labeled.

Bagging is a technique for generating a committee of learners by randomly sampling the initial training set and choosing subsets of examples. This method creates several initial training sets to initialize each of the learners in the committee. Query by bagging employs bagging to create the committee needed for the query by committee approach. We have adopted a version of this committee-based approach in order to reduce the number of user-labeled examples.

Figure 3.3 graphically shows the learning algorithm for the query by bagging technique. The first step is selecting a small initial set of training examples. The set of candidate mappings (Figure 2.7) serve as the set of examples from which the training examples are chosen. In order to choose the training examples for this initial training set, the candidate mappings are ranked according to their total object similarity scores (Figure 2.7). It is then determined for each transformation, across all attributes, which are the highest ranked candidate mappings that contain that transformation. A set of these candidate mappings will be chosen as examples and labeled for the initial training set.

Table 3.1 shows the highest ranked mapping for each transformation and each attribute. These mappings compose the set of examples from which the initial training set is drawn. This selection process insures that a variety of examples would be included because there are a sparse number of positive examples (true mappings) among the candidate mappings. Having a variety of examples is important for creating a diverse set of decision tree learners and for learning accurate transformation weights.

Transformations	Name Mapping	Street Mapping	Phone Mapping
Equality	1	1	1
Stemming	1	7	
Soundex	100	200	
Abbreviation	7	210	
Initial	3	232	77
Prefix	98	200	84
Suffix	2030	77	92
Substring		7	49
Acronym	67	200	
Abbreviation2	2030	56	

Initial Training Set: 1, 3, 7, 49, 56, 67, 77, 84, 92, 98, 100, 200, 210, 232, 2030

Table 3.1: Choosing the Initial Training Set

Once the initial training set has been created, the next step is to use the bagging technique to initialize the learners. Bagging randomly samples the initial training set, choosing subsets of examples to initialize each learner in the committee. Each decision tree learner is initialized with a different subset of the

initial training set. From these training examples the decision tree learner efficiently constructs a decision tree that determines the important attributes and thresholds for deciding a mapping. This decision tree is then converted into a set of mapping rules. These mapping rules are used to classify the remaining examples (candidate mappings) (Figure 2.7). If the similarity scores of a candidate mapping fulfill the conditions of a rule then the candidate mapping is classified as mapped.

3.1.3 Choosing the Next Example

With a committee of decision tree learners, the classification of an example or candidate mapping by one decision tree learner is considered its vote on the example. The votes of the committee of learners determine which examples are to be labeled by the user. One of the key factors in choosing an example is the disagreement of the query committee on its classification (Figure 3.4). The maximal disagreement occurs when there are an equal number of mapped (yes) and not mapped (no) votes on the classification of an example. This example has the highest guaranteed information gain because regardless of the example's label half of the committee will need to update their hypothesis. As shown in Figure 3.4 the example **CPK, California Pizza Kitchen** is the most informative example for the committee (**L1, L2, L3, L4, L5, L6, L7, L8, L9,** and **L10**).

Examples	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10
Art's Deli, Art's Delicatessen	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CPK, California Pizza Kitchen	Yes	No	Yes	No	Yes	Yes	Yes	No	No	No
Ca'Brea, La Brea Bakery	No	No	No	No	No	No	No	No	No	No

Figure 3.4: Committee Votes

The committee votes are used in deciding the next example to label, and they are also necessary for determining the classification of an example. Each learner votes on all candidate mappings. Those mappings where the majority of the learners vote *yes* are considered mapped otherwise they are not mapped. These classifications of the examples are then given as input to the transformation weight learner.

Both the mapping-rule learner and the transformation weight learner influence the decision on choosing the next example to be labeled by the user. The mapping-rule learner contributes the committee votes on each example, and the transformation weight learner provides the new ranking of the examples based on their total object similarity scores. Using the mapping-rule learner criteria allows for the example with the most information gain to be chosen, yet there are cases where the committee disagrees the most on several examples.

Figure 3.5 shows the order in which four different criteria can be applied in order to select one example from the set of candidate mappings. Because there are so few positive examples in the dataset the system prefers to increase the likelihood of choosing positive examples.

Select Highly Ranked Examples The first step is for the system to select out a small set of high quality examples from the complete set of examples computed by the candidate generator. This is accomplished by enforcing an at-most-one relationship on the set of mappings. This problem of finding an at-most-one mapping that maximizes the total mapping assignment can be viewed as a weighted bipartite matching assignment problem (Figure 3.6), where the total object similarity scores of each of the candidate mappings serve as weights. The learner can solve this problem using a technique called the Hungarian method [70].

Disagreement of Committee Votes The disagreement of committee votes, as described previously (Figure 3.4), is considered for each mapping of the set computed in step 1. The examples are ranked according to the disagreement of the committee votes. If there is only one example with the highest level of disagreement than that example is chosen for the user to label.

Dissimilarity to Previous Queries If there are a set of examples with the same disagreement on the committee votes then in order to reduce this

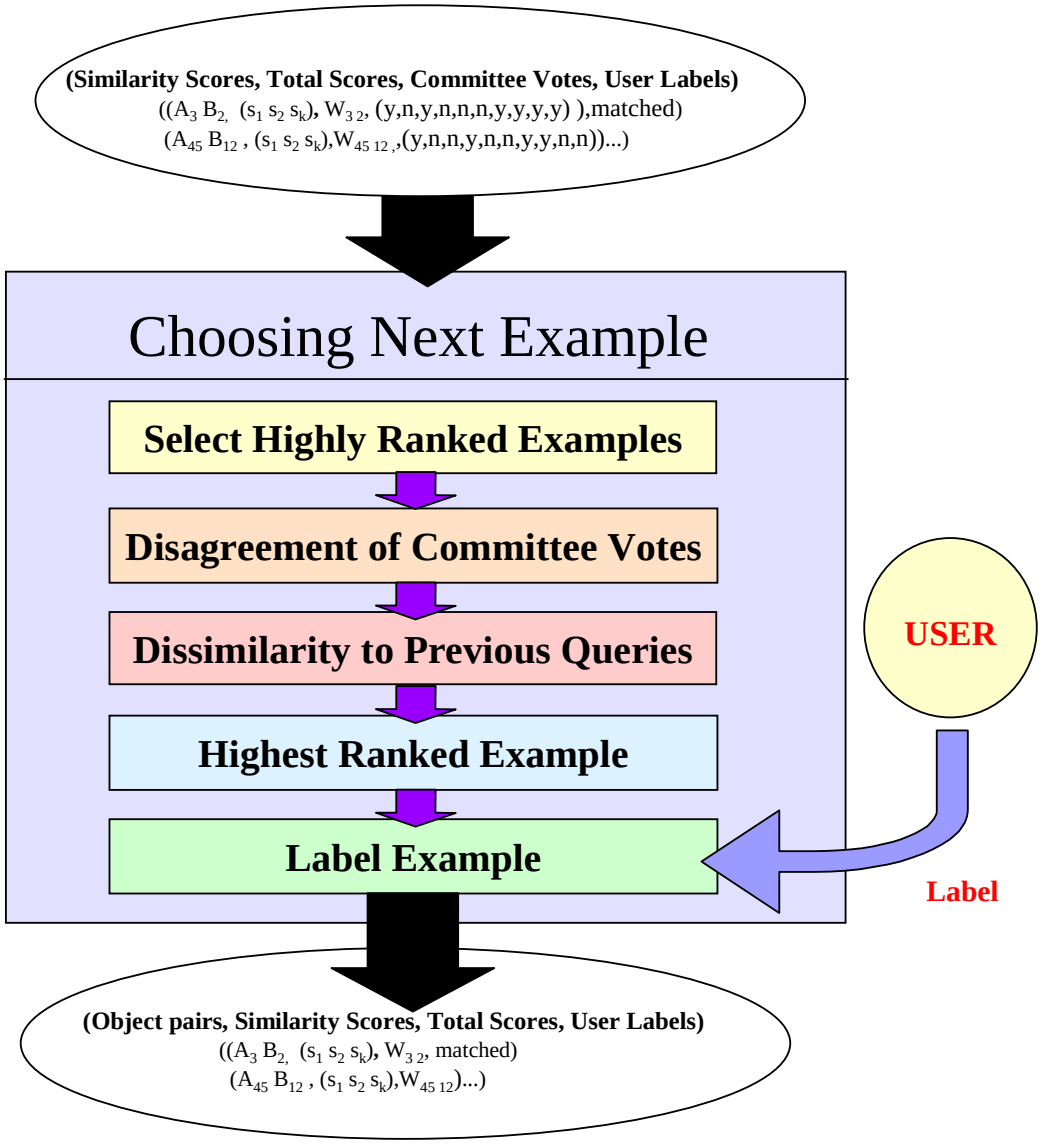

Figure 3.5: Choosing Next Example

set we examine these examples to choose the example that is most unlike or dissimilar to previous examples labeled by the user. The learner determines dissimilarity by applying the Autoclass clustering system to the

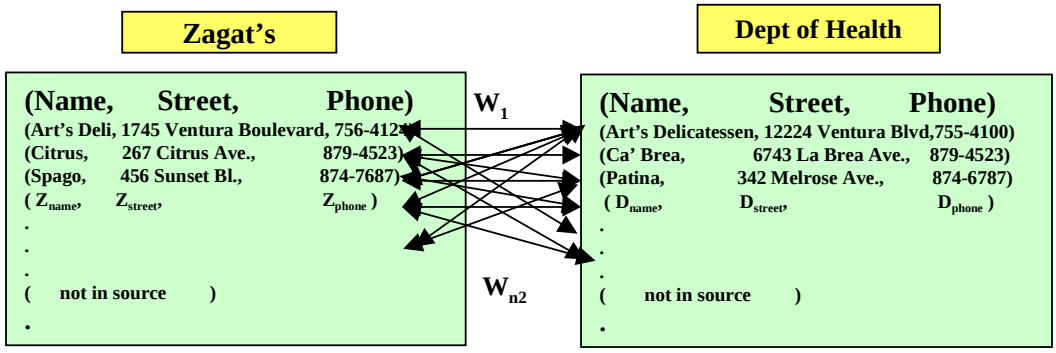

Figure 3.6: Object Identification as Weighted Bipartite Graph

initial candidate mappings. Similar examples will be in the same cluster, so the system keeps track of the clusters that previous labeled examples are in, in order to choose an example which is not in one of those clusters.

Highest Ranked Example If there are set of examples with the same dissimilarity, (i.e. from the same cluster) then they are ranked according to their total object similarity score. The highest ranked example is chosen from the set. Again, since there are a sparse number of positive examples in the set of candidate mappings, we would like to increase the chances of choosing a positive example for the committee to learn from.

Once an example is chosen, the user is asked to label the example. If it is known that there is an at-most-one relationship between the objects, then for each example positively labeled by the user the system automatically labels the remaining unlabeled examples containing the mapped objects as negative

examples. After the user labels the query example, the learner updates the committee and learns new mapping rules in order to reclassify the examples.

This learning process is repeated until either all learners in the committee converge to the same decision tree or the user threshold for labeling examples has been reached. When learning has ended, the mapping-rule learner outputs a majority-vote classifier that can be used to classify the remaining pairs as mapped or not mapped. At this point if it is known that there is an at-most-one relationship between the objects, then this relationship can be enforced in the remaining mappings positively classified. Because this problem of mapping two sets of objects can be viewed as a weighted bipartite matching assignment (Figure 3.6) problem the system uses the Hungarian method to find the maximal total matching assignment. After the total mapping assignment is complete, new information sources (mapping tables) can be created for use by an information mediator.

3.2 Transformation Weight Learning

The purpose of optimizing the transformation weights is to reduce the number of labeled examples needed by the system to achieve high accuracy mapping. The transformation weight learner must learn how to increase the similarity scores for the correct mappings, while decreasing the scores for the incorrect mappings.

Having the correct mapping scores higher than the incorrect mapping scores will allow the mapping-rule learner to construct higher accuracy decision trees with fewer labeled examples.

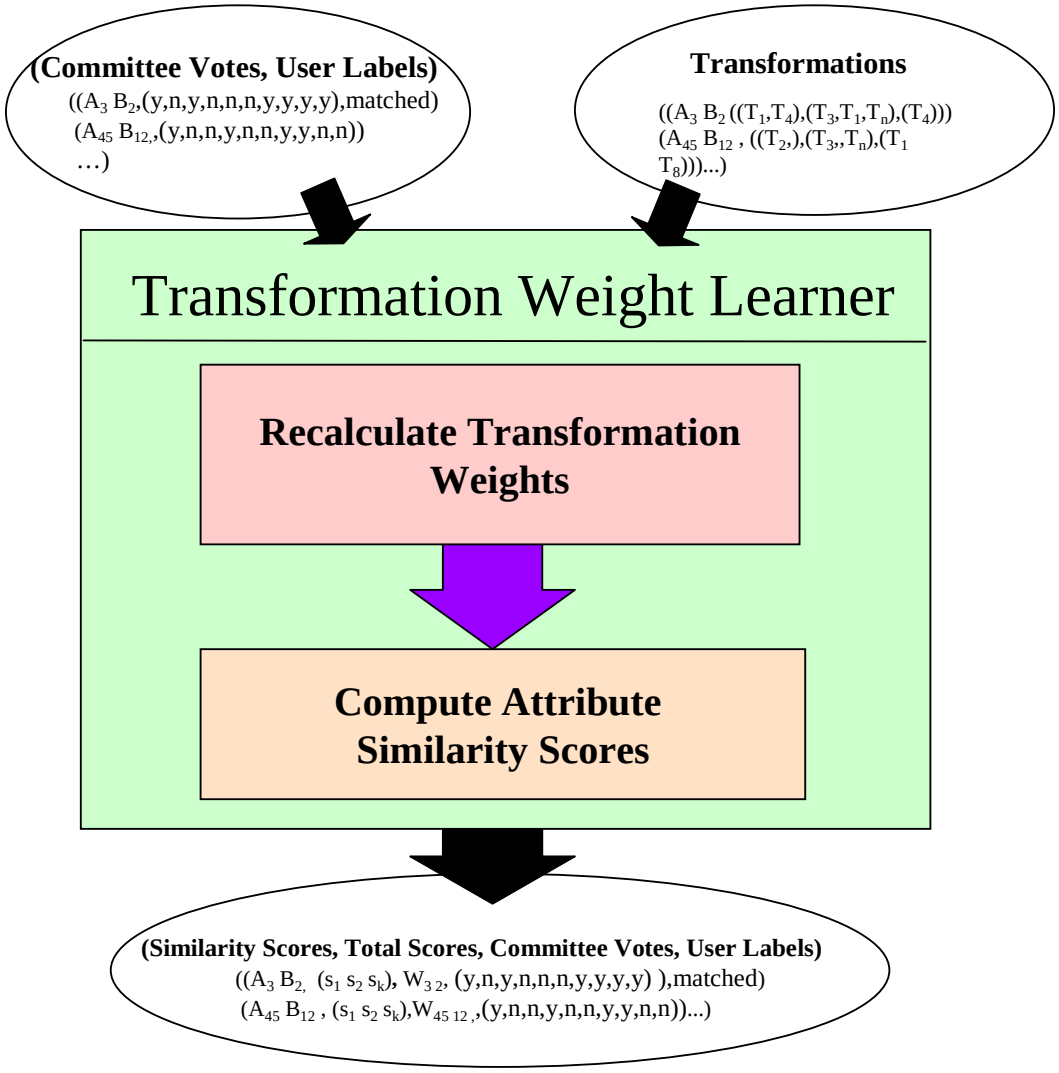

Figure 3.7: Transformation-Weight Learner

Given as input to the transformation weight learner are the mapping-rule learner's classifications of the examples or committee votes and the set of applied transformations for each example collected by the candidate generator. With this input the transformation weight learner first calculates the transformation weights for each transformation and then uses these new probability scores to recalculate the attribute similarity scores for each example. Once all of the attribute similarity scores are calculated for every attribute, the examples are then ranked according to their total object similarity scores.

3.2.1 Calculating Transformation Weights

The method for calculating the new transformation weights takes into consideration the classifications of both the user-labeled and unlabeled examples. The transformation weights measure the likelihood that a given transformation, like "Equality" or "Acronym," participates in a correct mapping between two objects. Tabel 3.2.1 shows the example input to the transformation weight learner from the mapping-rule learner.

Restaurant Mappings	Label	Labeled by
("Carnegie Deli","Trattoria Dell'Arte")	No	Learner
("Art's Delicatessen","Art's Deli")	Yes	Learner
("Spago","Spago (Los Angeles)")	No	Learner
("California Pizza Kitchen","CPK")	Yes	User
("Jan's Restaurant","Joe's Restaurant")	No	Learner

Table 3.2: Candidate Mappings with Classifications

Because initially it is not known which transformations are appropriate for the application domain, the initial attribute similarity scores determined by the candidate generator do not accurately represent the true similarity between all of the objects. Therefore, due to the inaccuracy of the initial attribute similarity scores, there is some error in the mapping rules and the classifications of the examples by the mapping-rule learner. While there are some misclassifications of the unlabeled data, they still help to increase the accuracy of the transformation weights. Using an active learning approach reduces the number of user-labeled data, so there are sparse number of labeled examples to learn the correct transformation weights from.

The unlabeled data augments the information about transformations known from the labeled examples. Yet, because there are misclassifications on the unlabeled examples they should not have the same impact on the computation of the transformation weights as the labeled examples, as discussed in previous work on combining labeled and unlabeled examples [68]. In order for the labeled examples to have greater importance than the unlabeled ones, the population of the labeled examples is increased by adding α duplicates of each of the labeled examples to the set of training data.

Given the classifications of the examples the transformation weights can be tailored to the application domain. The likelihood that the transformation t

will participate in a mapping $\hat{m}$ between two objects can be estimated using the following formula:

$$p(m \mid t_i) = \frac{positive\ classifications\ with\ transformation_i}{total\ number\ of\ transformation_i}$$

$p(m \mid t_i)$ is calculated for each of the general transformations detailed in Chapter 2, i.e. Equality, Acronym, etc. Therefore, the instantiated transformations, like (Equality "Art" "Art") and (Equality "s" "s"), of the general transformation Equality will have the same transformation weight, and the classifications of mappings which use these instantiated transformations will contribute to the calculation of the transformation weight of the general transformation, i.e. Equality. Once all of the transformation weights have been calculated then the attributes similarity scores are computed.

3.2.2 Re-computing Attribute Similarity scores

To determine the attribute similarity scores for each candidate mapping, first the product of the probabilities of the applied transformations is computed, and then normalized.

$$AttributeSimilarityScore(A, B) = \prod_{i=1}^{n} t_i$$

Given the set of transformations computed for the mapping by the candidate generator and the newly calculated transformation weights (Table 3.3) the attribute

similarity score is computed and then normalized. Table 3.3 shows an example of how the attribute similarity scores are recalculated for the candidate mapping of "Art's Deli" and " Art's Delicatessen." The computing of the attribute similarity scores is repeated for each attributes.

Example:

Mapping: "Art's Deli" and " Art's Delicatessen"

Transformation	p(m | t)	¬p(m | t)
(EQUAL "Art" "Art")	.9	.1
(EQUAL "s" "s")	.9	.1
(PREFIX "Deli" "Delicatessen")	.3	.7

Total mapped score m = .243
Total not mapped score n = .007

$$NormalizedAttributeSimilarityScore = \frac{m}{(m+n)}$$

$$= \frac{.243}{(.243 + .007)}$$

$$AttributeSimilarityScore = .9612$$

Table 3.3: Recalculating Attribute Similarity Scores

When the attribute similarity scores have been calculated then the total object similarity scores are again computed for each candidate mapping as shown in section 2. These candidate mappings are ranked according the new total object similarity scores. The new scores, attribute and object similarity scores, are given to the mapping-rule learner in order to create more accurate mapping rules. They

play an important factor in increasing the mapping accuracy and deciding the next example to be labeled.

Chapter 4

EXPERIMENTAL RESULTS

In this section we present the experimental results that we have obtained from running Active Atlas across three different application domains: Restaurants, Companies and Airports. Three different variations of Active Atlas have been created to compare against the full version of Active Atlas across all three domains. The first variation does not perform transformation weight learning, while the second does not use the dissimilarity of examples to previous queries to assist in choosing examples for the user to label. The third variation of Active Atlas does not enforce the at-most-one relationship on the mapping assignment.

We have included results from two baseline experiments as well. For each domain, we ran experiments for a system called Passive Atlas. The Passive Atlas system includes the candidate generator for proposing candidate mappings and a single C4.5 decision tree learner for learning the mapping rules. The second baseline experiment runs the candidate generator only and requires the user to review the ranked list of candidate mappings to choose an optimal mapping threshold. In this experiment only the stemming transformation is used, which is similar to an information retrieval system, such as Whirl [22]. We will refer to this experiment as the IR system.

Once the transformation weights and mapping rules have been tailored to a specific application domain we would like to be able to reuse this learned information in order to classify the mappings of new objects. This would avoid the expense of rerunning Active Atlas to learn to classify the new objects. The last set of experiments measure the accuracy of learned weights and rules on classifying new objects for each of the domains.

4.1 Restaurant Domain

For the restaurant domain, the shared object attributes are **Name, Street,** and **Phone.** Many of the data objects in this domain match almost exactly on all attributes, but there are types of examples that do not, as shown in Figure 1.1.

Because of these four types of examples, the system learns two mapping rules: if the restaurants match highly on the **Name** & **Street** or on the **Name** & **Phone** attributes then the objects should be mapped together. The example "Art's Deli" is an example of the first type of mapping rule because its **Name** and **Street** attribute values match highly. The "Les Celebrites" example is an example of the second type of mapping rule, because its **Name** and **Phone** attribute values match highly. These two mapping rules are used to classify all of the candidate mappings. Any candidate mapping that fulfills the conditions of these rules, will be mapped. Because in our application we are looking for the correct health rating of a specific restaurant, examples matching only on the **Name** attribute, like the "Teresa's" example, or only on the **Street** or **Phone** attribute, like "Steakhouse The" are not considered mapped.

4.1.1 Experimental Results

In this domain the Zagat's website has 331 objects and the Dept of Health has 533 objects. There are 112 correct mappings between the two sites. When running the IR system experiment, the system returns a ranked set of all the candidate mappings. The user must scan the mappings and decide on the mapping threshold or cutoff point in the returned ranked list. Every candidate mapping above the threshold is classified as mapped and every candidate mapping below the

threshold is not mapped. The optimal mapping threshold has the highest accuracy. Accuracy is measured as the total number of correct classifications of the candidate mappings divided by the sum of the total number of candidate mappings and the number of true object mappings not proposed. This method is comparable to the Whirl system [19].

Listed in the following table is the accuracy information at specific thresholds for the IR system experiment (Figure 4.1). The mapping threshold with the highest accuracy is highlighted. In Figure 4.1 the optimal mapping threshold is at rank 111 in the list, and therefore, the top 111 examples in the list are considered mapped together. The table shows that at this optimal threshold, only 109 examples of the 111 are correct mappings, 3 true examples have been missed and 2 false examples have been included; therefore, 5 examples in total are incorrectly classified. In this domain application, a threshold can not be chosen to achieve perfect accuracy. In general, selecting the optimal threshold to obtain the highest possible accuracy is an unsolved problem.

# of Ranked Examples	# of Correct Mappings	# of Missed Mappings	# of False Mappings	Accuracy
70	70	42	0	0.9873
92	91	21	1	0.9934
111	109	3	2	0.9985
116	110	2	6	0.9976
119	111	1	8	0.9973
128	112	0	16	0.9952

Figure 4.1: IR System Results

We compared the accuracy of the IR system results against the accuracy of the two learning systems. The purpose of the Passive Atlas experiments are to show that learning the mapping rules can achieve higher accuracy than the IR system experiment, while also demonstrating that Active Atlas can achieve the higher accuracy with fewer labeled examples. The goal of both learning systems is to deduce more information about how the objects match in order to increase the accuracy of the total mapping assignment. The results of these experiments are shown in Figure 4.2.

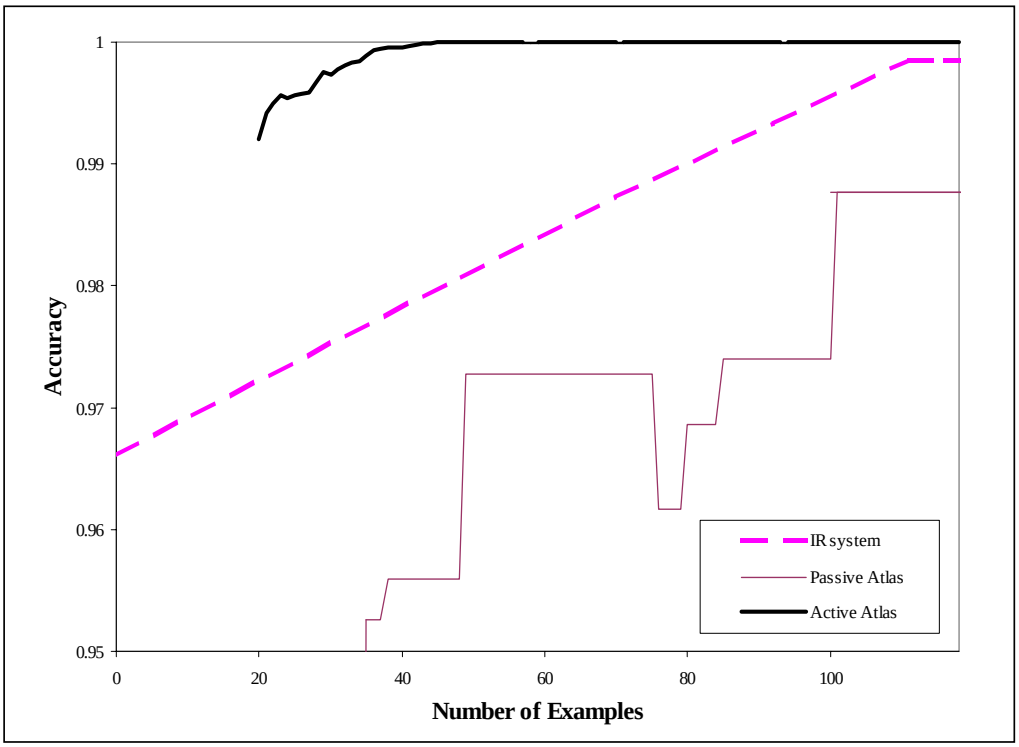

Figure 4.2: Restaurant Domain Experimental Results

The accuracy results from the three types of experiments are shown in relation to the number of examples that were labeled. For the two learning systems the results have been averaged over 10 runs, and the learners classified 3310 candidate mappings proposed by the candidate generator. Figure 4.2 shows that learning the mapping rules increases the accuracy of the mapping assignment. In the Active Atlas experiments, the system achieved 100% accuracy at 45 examples, while Passive Atlas surpassed the IR system results at 1594 and reached 100% accuracy at 2319 examples. The graph also shows that Active Atlas requires fewer labeled examples than Passive Atlas.

The active learner is able to outperform the passive learner because it is able to choose examples that give it the most information about the domain and guide the learning process. The passive learner chooses examples in a random manner, independent of the actual data. Because these domains have a sparse number of positive (mapped) examples, on the order of 1% of the data, it is harder for the passive learner to randomly choose positive examples that lead to high accuracy mapping rules; and therefore, it requires more examples.

Another factor in the dramatic improvement of Active Atlas over the these two baseline experiments is transformation weight learning. The same set of general transformations is applied in every application domain and for every attribute. These transformations can suggest possible relationships between tokens, e.g. (Abbreviation "Deli", "Delicatessen") or between phrases, e.g. (Acronym

"California Pizza Kitchen", "CPK"), but these relationships may not accurately represent the true relationship between the tokens. Therefore, the initial attribute similarity scores calculated by the candidate generator may inaccurately reflect the similarity of the attribute values. The two baseline approaches must classify examples with inaccurate similarity scores, while Active Atlas can learn to adjust the similarity scores to more accurately capture the true similarity between the attribute values.

Figure 4.3 presents the results of the variations of Active Atlas. There are four experimental results shown in the figure: Active Atlas as shown in Figure 4.2, Active Atlas without the transformation weight learning, Active Atlas without dissimilarity for choosing the next example, and Active Atlas without enforcing the at-most-one relationship between the objects. For all Active Atlas experiments there were 10 learners on the committee and 20 user-labeled initial training examples chosen by the system to optimize the transformation weight learning.

The full Active Atlas system achieves 100% accuracy using the fewest number of examples (45). Active Active Atlas without dissimilarity achieves 100% accuracy with 192 examples. Using the dissimilarity of examples to previous queries has more influence in the Airport domain. Atlas without the transformation weight learning takes 289 examples to achieve 100% accuracy. In the Restaurant domain there are fewer text inconsistencies than the other two domains,

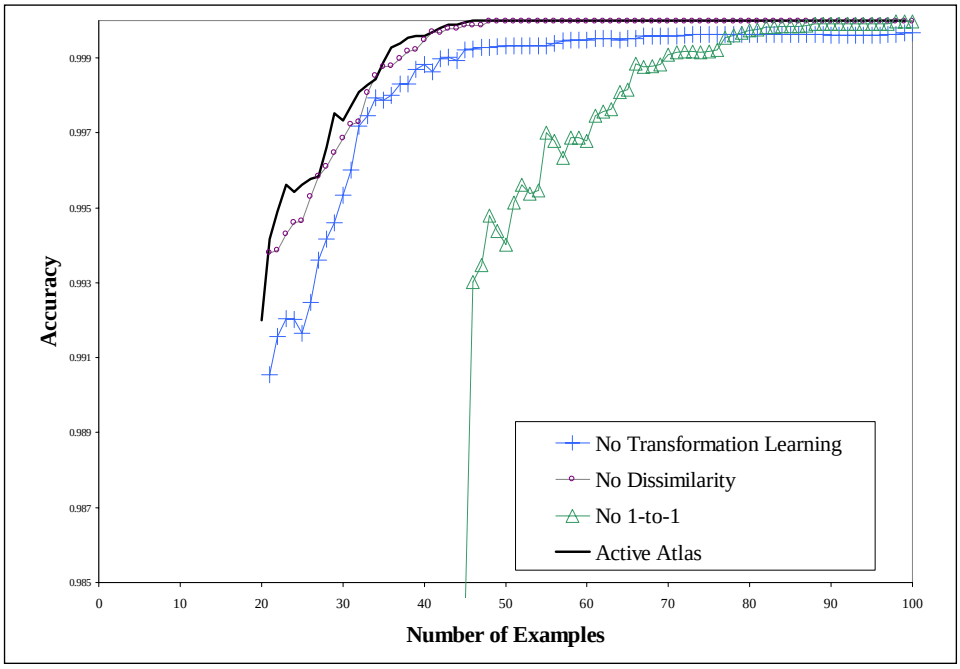

Figure 4.3: Restaurant Domain Active Atlas Results

and therefore, there are fewer types of transformations applied by the candidate generator. This makes learning the transformation weights possible with fewer examples.

Also, in this domain when the clustering algorithm is applied to the examples to determine their similarity, the majority of positive examples are contained in one cluster, meaning that majority of positive examples are similar to each other. The main advantage of using dissimilarity is to catch the unusual examples that might not have high total object similarity scores and that are contained in a variety of clusters.

Active Atlas without enforcing the at-most-one relationship achieves 100% accuracy at 343 examples. It initially has difficulty classifying the examples as shown in Figure 4.3, where it took 45 examples to achieve accuracy comparable to the other versions of Active Atlas (i.e., greater than .98). This is because without the at-most-one enforcement the initial training set chosen to optimize the transformation weight learning consists almost completely of positive examples. With the at-most-one enforcement for each positive example labeled by the user all other examples which proposed between either of the mapped objects will be labeled as a negative examples by the system. Therefore, the committee of learners would be initialized with a large number of negative examples as well and the transformation weights can be more accurately computed. The initial examples impacts this domain more than the other two domains again because there are only a few types of transformations applied in this domain.

4.2 Company Domain

In the company domain there are two websites, HooversWeb and IonTech, which both contain information on companies (**Name, Url** and **Description**). This domain was provided by William Cohen [19]. In this domain the **Url** attribute is usually a very good indicator for companies to match on, e.g. "Soundworks" (Figure 4.4). There are examples where the **Name** matches very well, but the

Url is not an exact match ("Cheyenne Software"); or, where the **Url** matches exactly, but the names are not matched at all ("Alpharel" & "Altris Software"). Atlas, therefore, learns the thresholds on the combination of the attributes **Name** and **Url**, where one attribute needs to be highly matched and the other partially matched in order for there to be a mapping between the objects.

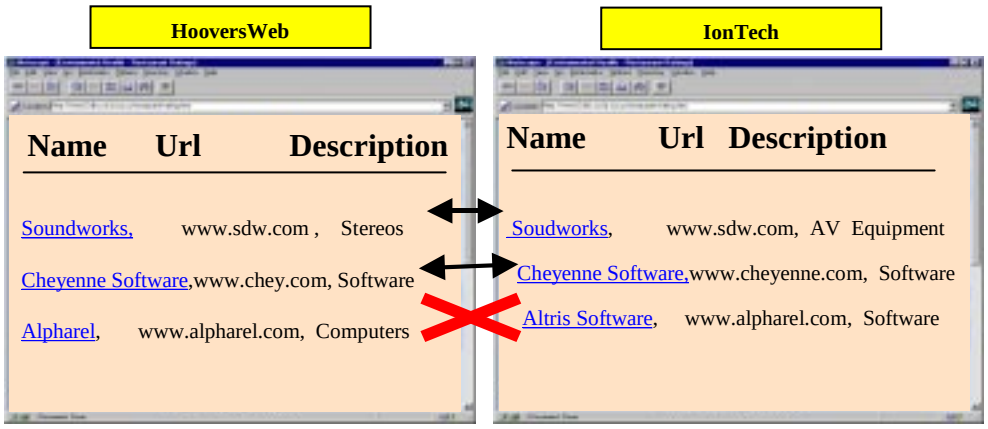

Figure 4.4: Company Domain Examples

4.2.1 Experimental Results

In this domain HooversWeb has 1163 objects and the IonTech site has 957 objects. There are 294 correct mappings between the sites. The results for the IR system case are shown in Figure 4.5. The optimal threshold is at rank 282, where 20 examples are incorrectly classified and 5% of the object mappings are missing.

Ranked Examples	Correct Mappings	Missed Mappings	False Mappings	Accuracy
50	50	244	0	0.9829
223	222	72	1	0.9949
282	278	16	4	0.9986
396	287	7	82	0.9938
4236	294	0	3942	0.7244

Figure 4.5: IR System Results

Figure 4.6 shows the results for the learning experiments. For these experiments in the company domain, the candidate generator proposed 14303 candidate mappings, and the results have been averaged over 10 runs. Similar to the Restaurant domain, Figure 4.6 shows that learning the mapping rules increases the accuracy of the mapping assignment. Active Atlas achieves higher accuracy than the IR system experiments immediately (at 7120 examples for Passive Atlas). The graph clearly demonstrates that the active learner requires fewer labeled examples than Passive Atlas, as well as demonstrating the effect of the inaccuracies of the initial attribute similarity scores on being able to classify the mappings.

The transformations have more influence in this domain. The transformations are able to resolve the spelling mistake between "Soundworks" and "Soudworks" (Figure 4.4) using the Soundex transformation, which made the difference in it being mapped or unmapped. In the Active Atlas experiments, the system achieved

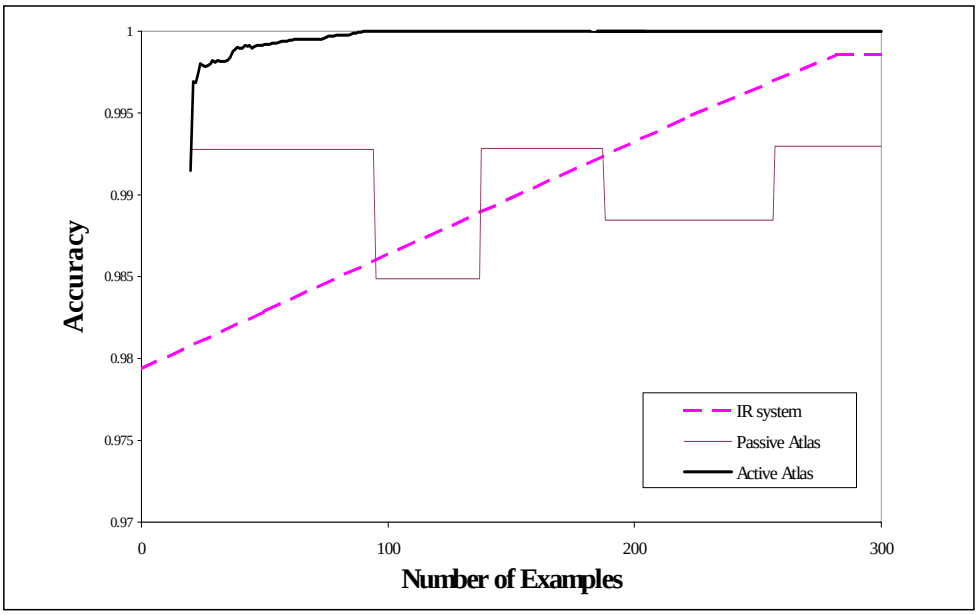

Figure 4.6: Company Domain Experimental Results

100% accuracy at 95 examples (Figure 4.7). Active Atlas without the transformation weight learning takes the most examples to achieve 100% accuracy. Using the dissimilarity of the examples has more impact in this domain. The variation of Active Atlas which did not use dissimilarity achieved 100% accuracy at 142 examples. Without using dissimilarity to assist in choosing examples the system labels more examples in order to choose examples that the committee disagrees equally on, but have lower total object similarity scores.

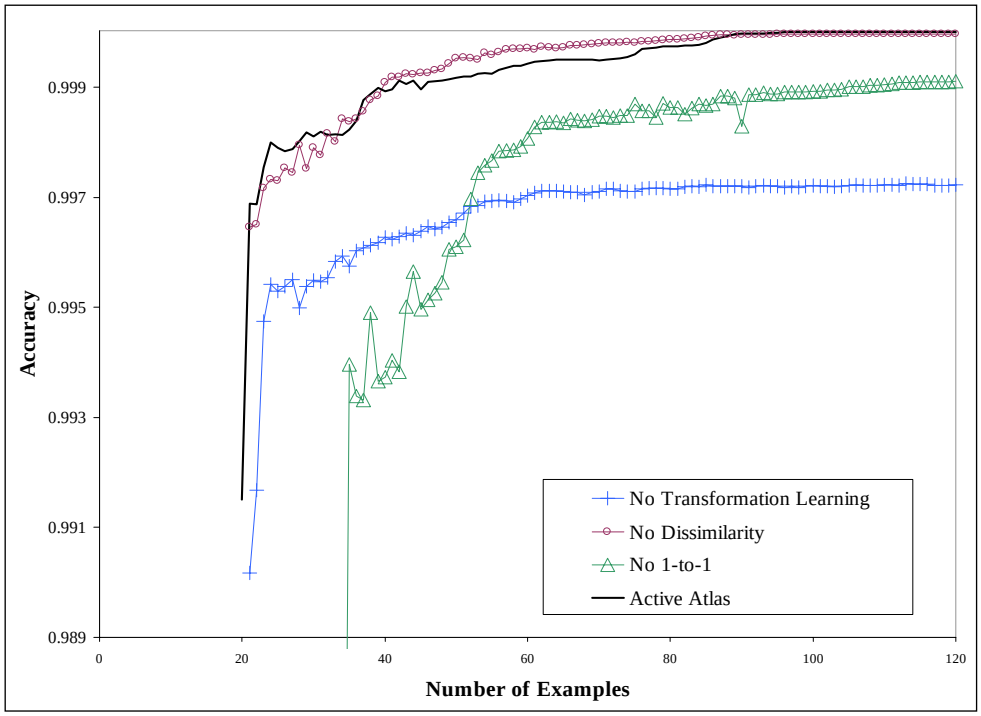

Figure 4.7: Company Domain Active Atlas Results

4.3 Airport/Weather Domain

We have a list of 428 airports in the United States and a list of over 12,000 weather stations in the world. In order to determine the weather at each airport, we would like to map each airport with its weather station. The airports and the weather stations share two attributes (**Code** and **Location**). The airport code is a three letter code (e.g., ADQ), and the weather station code is a four letter code (e.g., PADQ). In the majority of examples the airport code is the last three letters of the weather station code, like the "Kodiak" example in Figure 4.8.

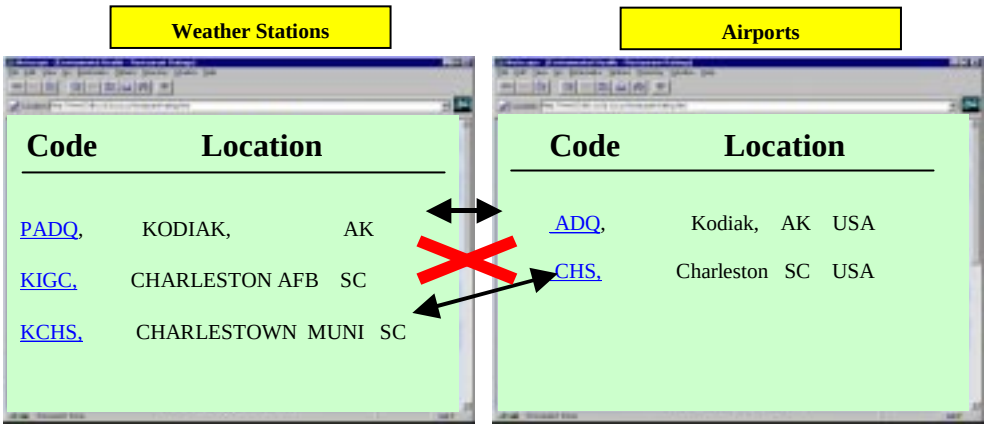

Figure 4.8: Airport/Weather Domain examples

4.3.1 Experimental Results

The results in this domain for the IR system at the selected threshold are shown in Figure 4.9. The optimal threshold is set at rank 438, where 220 examples are incorrectly classified and 24% of the object mappings are missing.

Ranked Examples	Correct Mappings	Missed Mappings	False Mappings	Accuracy
19	19	399	0	0.9767
355	276	142	79	0.9870
438	318	100	120	0.9871
479	331	87	148	0.9852
1667	400	18	1267	0.9249

Figure 4.9: IR System Results

In this domain the set of transformations plays a larger role in increasing the accuracy of the object mappings, as clearly shown by the IR system results. The

main reason for the lower accuracy of the experiments with stemming is because the IR system is not able to recognize that the airport code is a substring of the weather code for the **Code** attribute. It, therefore, only uses the **Location** attribute to match objects, so it makes mistakes, like mapping the "KIGC" and "CHS" objects. There are 18 object mappings that were not proposed by the candidate generator because it did not have the necessary transformations.

Like the other domains, Figure 4.10 shows that learning the mapping rules increases the accuracy of the mapping assignment. Active Atlas achieves 100% accuracy at 294 examples, and Passive Atlas achieves higher accuracy than the IR system results after only 80 examples.

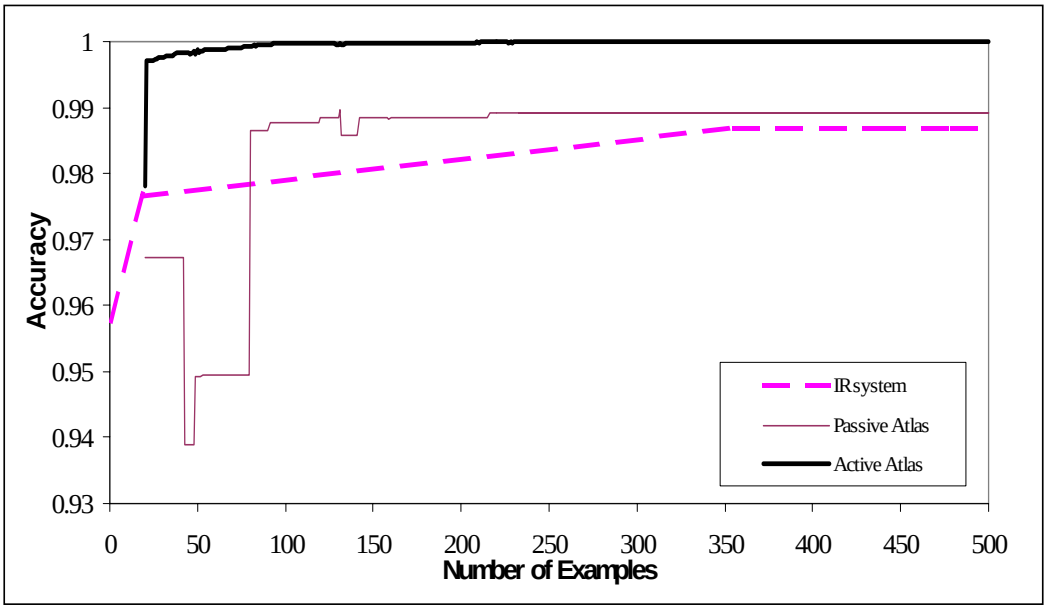

Figure 4.10: Airport/Weather Domain Experimental Results

Figure 4.11 presents the Active Atlas experiments. The dissimilarity of examples and the at-most-one relationship features have more impact on the systems accuracy in this domain. The majority of positive examples match highly on the **Code** attribute, yet approximately 25% of the positive examples do not match on the **Code** attribute or have missing **Code** attribute values. These positive examples would have attribute similarity code similar to the negative example "KIGC" and "CHS" in Figure 4.8. Using dissimilarity of examples assists in finding these unusual positive examples, while enforcing a at-most-one relationship reduces the negative examples, like "KIGC" and "CHS."

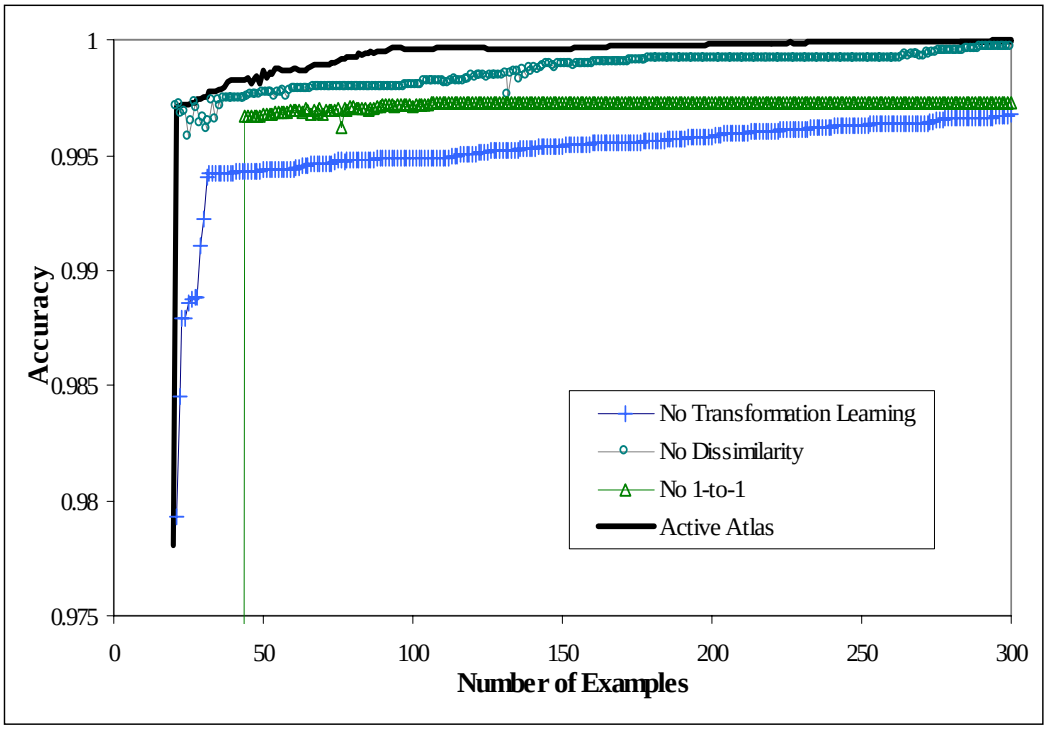

Figure 4.11: Airport/Weather Active Atlas Results

4.4 Applying Learned Weights and Rules to New Sources

Once the transformation weights and mapping rules have been tailored to a specific application domain we would like to be able to reuse this learned information in order to classify the mappings of new objects. There are two specific cases of information integration in which applying learned weights and rules would be desired. The first case is when the information sources (e.g., Zagat's and the Health Dept) for which the weights and rules were learned are updated with new objects, i.e. restaurants, and the second case is when a new information source with a similar domain is to be integrated into the application, such as the CuisineNet website which lists restaurants.

Being able to reuse the learned weights and rules to achieve high accuracy mapping of unseen objects would avoid the cost of rerunning Active Atlas to learn to classify the new objects. In order to measure how well weights and rules learned by Active Atlas classify unseen data, we performed a set of 5-fold cross-validation experiments for all three domains (Table 4.1). For each domain the set of training examples was partitioned into five sets. Active Atlas was trained on four sets of examples and tested on the fifth unseen set of examples. This was repeated so that each subset of examples was used as a test set.

Domain	Total Number of Examples	Number of Test Examples	Average Accuracy
Restaurant	3310	662	.9968
Company	14303	2861	.9980
Airport	17120	3624	.9951

Table 4.1: Accuracy of Learned Weights and Rules on Unseen Data

Table 4.1 shows the results of these experiments averaged over 5 complete runs for each domain. The experimental results demonstrate that applying the learned weights and rules to unseen data from a similar domain can achieve high accuracy mapping of objects.

Chapter 5

RELATED WORK

Previous work on object identification has either employed manual methods to customize rules for each domain or has required the user to apply a fixed threshold to determine which objects are considered mapped together. These systems generally require heavy user interaction in order to achieve high accuracy mapping. None of the previous work apply general domain-independent transformations that are then adjusted to fit the specific application domain. The main advantage of our system is that it can, with high accuracy, learn to simultaneously

tailor mapping rules and transformations to a specific domain, while limiting user involvement.

The object identification problem occurs as a part of larger application areas, such as multi-source integration, automatic domain modeling, and data warehousing. There are four solution approaches that included the problem of object identification (Figure 5.1): databases, information retrieval, sensor fusion, and record linkage.

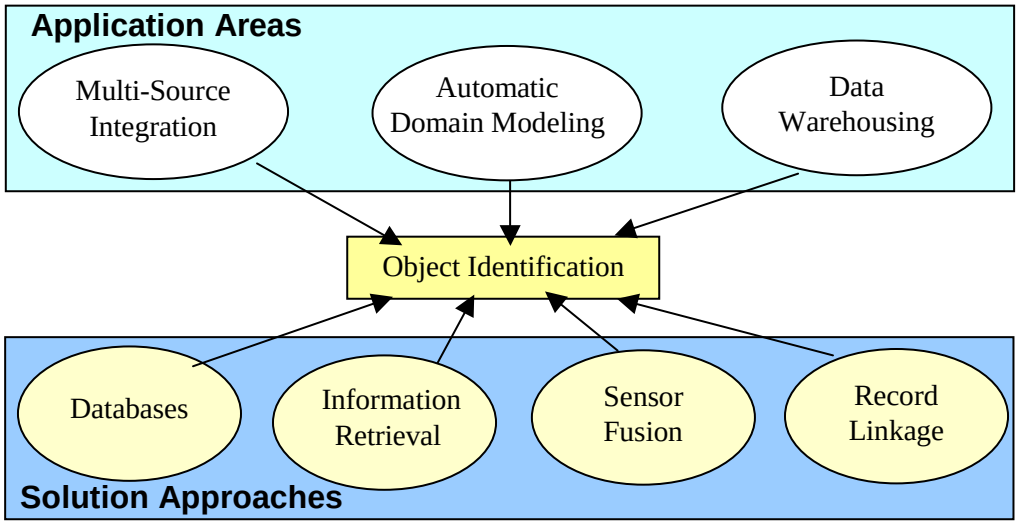

Figure 5.1: Related work graph

5.1 Application Areas

Research in multi-source integration [86] is concerned with dynamically and efficiently accessing, retrieving and integrating information from multiple information sources. The problem of object identification can appear when integrating information from sources that use different formatting conventions [6, 36, 81]. General information integration systems, like Pegasus [2], TSIMMIS [33], Infomaster [34], InfoSleuth [7], COIN [13], and Information Manifold [54], were developed to manage objects from multiple information sources, but they do not offer a general method to determine the mapping of objects which contain text formatting differences. There are a few general systems [72, 71, 91, 48, 26] that allow for user-defined domain-specific functions to determine how to map objects with formatting differences.

Current work on the problem of automatic domain modeling or schema integration [24, 25, 55, 63, 79] focuses on generating a global model of the domain for single, as well as multiple information sources. Domain models contain information about the relationships in the data. To create an abstract model or set of classification rules, these domain modeling tools compare the data objects in the sources using statistical measurements. When the information sources contain text formatting differences, these tools are not able to generate the correct domain model because the data objects cannot be mapped properly. The text

formatting differences must be resolved before using the domain modeling tools or capabilities to handle these formatting differences. In this context the objection identification problem can be considered a data cleaning technique [27].

Data warehousing creates a repository of data retrieved and integrated from multiple information sources given a common domain model or global schema. Research on data warehousing concentrates on efficient methods to merge and maintain the data [38, 83, 14, 15]. When merging the data from the target information sources, problems can occur if there are errors and inconsistencies in the data. To improve the integrity of the data warehouse, the data can be first filtered or cleaned before the repository is created [84]. Object identification techniques can be applied here to assist in cleaning the data [32].

5.2 Solution Approaches

The following solution approaches for object identification were developed by the database, information retrieval and statistical communities.

5.2.1 Databases

In the database community the problem of object identification is also known as the merge/purge problem, a form of data cleaning. Domain-specific techniques for correcting format inconsistencies [17, 9, 10, 37, 51, 75] have been applied by

many object identification systems to measure text similarity [32, 31, 41, 39, 40, 46, 90]. The main concern with domain specific transformation rules is that it is very expensive, in terms of user involvement, to generate a set of comprehensive rules that are specific not only to the domain, but also to the data in the two information sources that are being integrated.

There are also approaches [66, 65, 64, 73, 74] that use a single very general transformation function, like edit distance, to address the format inconsistencies. In our research we show that having a single general transformation function is not flexible enough to optimize the mappings across several domains. The same format inconsistencies can occur in many different domains, but they may not be appropriate or correct for all of those domains.

Recent work by Hernandez and Stolfo [41, 39, 40], Lu et al [53, 60, 61], Chatterjee and Segev [16], and Pu [77] have used mapping rules or templates to determine mapped objects, but these rules are hand tailored to each specific domain or limited to only the primary key. Work conducted by Pinheiro and Sun [73, 74] and Ganesh et al [32, 31] used supervised learning techniques to learn which combinations of attributes are important to generate a mapping. Both of these techniques assume that most objects in the data have at least one duplicate or matching object. They also require that the user provides positive examples of mapped objects, and in some cases the user labels as much as 50% of the data.

These techniques are most similar to the passive Atlas system, because they require either that the user choose the examples or they are randomly selected.

Another form of object identification focuses on comparing objects using attribute value conflict resolution rules [48, 59, 57, 56, 58, 85, 14, 15]. These rules do not judge attributes by their textual similarities, but by knowledge that they contain about how to compare attribute values when there is a conflict. For example, an object from one source may have the attribute **Age**, which holds the value for the age of a person, and an object from the other source may have the attribute **BirthDate**, which holds the value for the person's birthday. If there was an attribute value conflict resolution rule that could convert the **BirthDate** to **Age**, then the attribute values could be compared. This type of domain-specific data translation rule would be helpful for object identification, but is not the focus of our work. In our system a user-defined transformation function must be added for the two attributes **BirthDate** and **Age** to be correctly compared.

5.2.2 Information Retrieval

The problem of object identification has also appeared in the information retrieval community [80, 69]. When determining relevant documents to satisfy a user's query, words or tokens from the documents are compared. If there are text formatting differences in the documents, then relevant documents can be

missed. Closely related work on the Whirl object identification system by Cohen [20, 21, 18, 19, 23] views data objects from information sources as short documents. In this work the object mappings are determined by using the information retrieval vector space model to perform similarity joins on the shared attributes. A single transformation Stemming [76] is the only transformation used to calculate similarity between strings; therefore, in Figure 2.4 "CPK" would not match "California Pizza Kitchen." The similarity scores from each of the shared attributes are multiplied together in order to calculate the total similarity score of a possible mapping. This requires that objects must match well on all attributes. The total similarity scores are then ranked and the user is required to set a threshold determining the set of mappings. To set the threshold the user scans the ranked set of objects [19, page 9]. Setting a threshold to obtain optimal accuracy is not always a straightforward task for the user.

Work by Huffman and Steier [45, 44] on integrating information sources for an information retrieval system, uses domain-specific normalization techniques to convert data objects into a standard form for performing the mapping. This work involves maintaining a dictionary of domain-specific normalizations to apply when appropriate.

The Citeseer project [11, 35, 52] is a web-based information agent that finds relevant and similar papers and articles, where similarity is based on the set of citations listed in the articles. In order to determine identical citations, CiteSeer

first uses citation-normalization techniques to standardize article citations. The Identical Citation Grouping (ICG) method is then applied to group the identical citations. CiteSeer performs this method on the entire collection of citations, so that the similarity between the papers can be pre-computed. The ICG method focuses on addressing the addition and omission of words from the citations when measuring their similarity. Therefore, this method uses only one type of text transformation, i.e. "Equality," to compare the words and phrase of the citations.

Work conducted by McCallum et al [62] used a two step process of applying transformations in their approach to performing object identification on citations. Similar to our method in that they first apply less computationally expensive transformations to determine the initial set of mappings and then apply the more expensive transformation, edit distance, to compute the similarity metric between the objects. This requires the user to manually set the transformation weights for each new domain application.

5.2.3 Sensor Fusion

Related work conducted by Huang and Russell [42, 43] on mapping objects across information sources uses a probabilistic appearance model. Their approach also compares the objects based on all of the shared attributes. To determine similarity between two objects, they calculate the probability that given one object it

will appear like the second object in the other relation. Calculating these probabilities requires a training set of correctly paired objects (like the objects in the Figure 2.1). Unfortunately, appearance probabilities will not be helpful for an attribute with a unique set of objects, like restaurant **Name**. Since "Art's Deli" only occurs once in the set of objects, knowing that it appears like "Art's Delicatessen" does not help in mapping any other objects, like "CPK" and "California Pizza Kitchen."

In our approach we use a general set of transformations to handle this problem. The transformations are able to compare the textual similarity of two attribute values independent of other attribute value matches. If the transformation exists between the two attribute values, e.g. (Abbreviation - "Deli" "Delicatessen"), then it has a transformation weight associated with it. Because transformation weights should reflect the importance of the transformation for the matching of the attribute values, Active Atlas learns to adjust these weights to fit the specific domain.

5.2.4 Record Linkage

Probabilistic models of the data are also used within the record linkage community [3, 28, 47, 67, 89, 87]. Work in the record linkage community grew from the need to integrate government census data; therefore, they have developed domain-specific transformations for handling names and addresses. In a record

linkage system, the user is required to make several initial passes reviewing the data in order to improve and verify the accuracy of the transformations. Once the user is satisfied with the accuracy of the transformations, "blocking" attributes are chosen. Choosing blocking attributes is a way to reduce the set of candidate mappings by only including the pairs that match on the chosen attributes. The EM algorithm is then applied to learn the attribute weightings and classify the candidate mappings into one of three classes: mapped, not mapped, or to be reviewed by the user.

The main problem that the record linkage community [88, 47] found with the use of the EM algorithm is that because it is an unsupervised learning technique it may not divide the data into the desired classes. This problem does not occur with our approach because we incorporate the user's input into the learning process, as opposed to after the learning has completed. Another key difference between this approach and ours is that Active Atlas begins with set of domain-independent transformations and learns to tailor them to the specific domain, while the record linkage approach uses a set of static domain-specific transformations.

Chapter 6

CONCLUSIONS

For handling the problem of object identification there are two types of knowledge necessary: (1) the importance of the different attributes for deciding a mapping, and (2) the text formatting differences or transformations that may be relevant to the application domain. It is very expensive, in terms of the user's time, to manually encode these types of knowledge for an object identification system. Also, due to errors that can occur in the data, a user may not be able to provide comprehensive information without thoroughly reviewing the data in all sources.

We have adopted a general domain-independent approach for incorporating the user's knowledge into the object identification system. The Active Atlas system employs an object identification method that learns the mapping information necessary to properly integrate web sources with high accuracy. To achieve high accuracy object identification Active Atlas simultaneously learns to tailor both domain-independent transformations and mapping rules to a specific application domain through limited user input.

As shown in Figure 1.8, there are two stages in this method, computing the attribute similarity scores and learning objection identification rules and transformations to properly map the objects between the sources. In the first stage the candidate generator employs several information retrieval techniques to apply a general set of domain-independent transformations to the datasets. It uses these transformations to propose the set of possible mappings between the two sets of objects by comparing the attribute values and compute the similarity scores for the proposed mappings.

In the next stage the mapping learner determines which of the proposed mappings are correct. The mapping learner learns to tailor the appropriate mapping rules and transformations to the application domain. The mapping-rule learner computes which attributes or combination of attributes are appropriate for the specific application domain based on the attribute similarity scores and limited

input from the user. With this information the learner can accurately classify the proposed mappings.

Some transformations may also be more appropriate for a specific application domain than others. This error or bias in token relationships proposed by the transformations is not reflected in the initial attribute similarity scores calculated by the candidate generator. Therefore, in order to increase the accuracy of the similarity scores, as well as the mapping accuracy, it is important for the transformation weight learner to calculate domain appropriate weights for the transformations.

The mapping learner combines both types of learning to iteratively refine the mapping information necessary to accurately map the two sets of objects. As shown by the experimental results Active Atlas was able to achieve 100% accuracy while minimizing user involvement.

Some of the limitations of this work are linked to the assumptions that we have made about the nature of the problem of object identification. We have assumed that user would be able to give accurate labels to the example mappings chosen as queries by the learner, but this is not always the case as described further in the future work section. Another assumption is that the correct attribute information would be extracted from the websites for each object. Depending on the domain extracting the attribute information can be difficult. This is the main reason that the CiteSeer system compares objects as a whole instead of dividing

citations into attributes. The attributes of citations may not appear in the same order nor are the delimiters between attributes standard.

Other limitations of our system is that it requires the user to align the attributes that are to be compared. Future work on automatically aligning the attributes is needed. The problem of granularity can occur in a variety of domains. A few examples from the restaurant domain are that some restaurants would list there cuisine type as "Asian" while another would list "Japanese" or the city would be "Hollywood" while others would list it as "Los Angeles." The attribute values being compared do not share textual similarity and have different levels of granularity. A method to partially address this issue is described below.

6.1 Future Work

There are several avenues and issues for future work. Three of the more immediate issues that we are planning to address are: Noise or error in the labels provided by the user, learning domain-specific transformations weights, and learning to generate new transformations.

6.1.1 Noise in User Labels

When the user is asked for input by the mapping-rule learner the system, the user is presented with a single mapping of two objects. The user then labels the

example as mapped or not mapped. Upon being presented with the following query the user might be tempted to label it as mapped.

"Ritz-Carlton Cafe (Atlanta) 181 Peachtree 404-659-0400"
"Restaurant Ritz-Carlton Atlanta 181 Peachtree St. 404/659-0400"

Yet, the following example is the correct mapping for the objects for which the user may not ever be asked to label.

"Ritz-Carlton Restaurant 181 Peachtree St. 404-659-0400"
"Restaurant Ritz-Carlton Atlanta 181 Peachtree St. 404/659-0400"

In the company domain there are these two examples for the company "Gensym" which may cause some error on the part of the user:
Example pair 1:
"Gensym Corporation
http://www.gensym.com Computers/miscellaneous software"

"GENSYM CORP
http://www.gensym.com Software -Industrial-Scientific-Govt"

Example pair 2:

"Gensym Corporation

http://www.gensym.com Computers/ miscellaneous software"

"GENSYM CORP

http://www.gensym.com Software - Software-Solutions"

To practically address this problem of accurately labeling examples, Active Atlas could present the user with multiple queries to label at the same time. This will help to increase accuracy and reduce labeling confusion. Future work will focus on how to efficiently choose a set of queries while still minimizing user involvement.

6.1.2 Learning Specific Transformations Weights

The two restaurant objects shown below have been mapped together by Active Atlas. The set of transformations describing the relationship between the two restaurant's names is: (Equal "Katsu","Katsu") and (Drop "Restaurant"). The transformation weight learner only calculates the weights for the general transformations, so that the specific transformations (Equal "Katsu","Katsu") and (Equal "Art","Art") have the same transformation weight. This is because these specific transformations will occur only once in all of the set of mappings. Due

to the small number of labeled examples and the error in the mappings classified by the mapping-rule learner, the transformation weights for these specific transformations can not be reliably calculated.

"Restaurant Katsu 1972 N. Hillhurst Ave. 213/665-1891 Asian"

"Katsu 1972 Hillhurst Ave. 213-665-1891 Japanese"

Yet, there are some specific transformations that occur frequently in the data, such as (Drop "Restaurant"). This specific transformation occurs several times for positive mappings, and therefore should have a greater transformation weight than the general transformation Drop. Future work will be needed to determine how to choose which specific transformations weights should be calculated.

6.1.3 Learning to Generate New Transformations

In the example mapping shown above for the restaurant "Katsu," the words "Asian" and "Japanese" for the cuisine type do not have any general transformation relating them, but they co-occur for many other positively classified restaurant mappings. Learning a transformation that relates the cuisine types "Asian" and "Japanese" can help in classifying new restaurant objects that may

be added later to the website or when mapping the objects from a new information source. The issue for future work will be to develop a method for learning when to create these new types of transformations.

6.1.4 Long Term Issues

These are some of the long term issues that we plan to address in the future:

- Scaling: Approach currently applied to sets of examples on the order of 10,000. What are the issues for millions of examples?
- Applying Active Atlas to other types of related research problems, such as sensor fusion, objection identification for images, or creating bilingual corpora for machine translation
- Reconciling textual differences: Active Atlas recognizes differences and makes mappings between objects, but each object may still contain potentially conflicting information, e.g. the conflicting address information for the restaurant "Les Celebrites" (Figure 1.1). This problem is related to tracking objects over time. The restaurant "Les Celebrites" has changed locations, keeping track of this restaurant over time as more sources and information is added can help in developing a method for the system to properly reconcile the textual differences.

Reference List

[1] Naoki Abe and Hiroshi Mamitsuka. Query learning strategies using boosting and bagging. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

[2] R. Ahmed, P.D. Smedt, W.M. Du, W. Kent, M. Ketabchi, W.A. Litwin, A. Rafii, and M.C. Shan. The pegasus heterogeneous multidatabase system. *IEEE Computer*, 24:19–27, December 1991.

[3] W. Alvey and B. Jamerson. Record linkage techniques. In *Proceedings of an International Workshop and Exposition*, in Arlington, Virginia, Washington, DC. Federal Committee on Statistical Methodology, Office of Management and Budget, 1997.

[4] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[5] Yigal Arens, Chin Y. Chee, Chun-Nan Hsu, and Craig A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[6] C. Batini, M. Lenzerini, , and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surverys*, 18(4):323–364, December 1986.

[7] Roberto Bayardo, Jr., Bill Bohrer, Richard S. Brice, Andrzej Cichocki, Jerry Fowler, Abdelsalam Helal, Vipul Kashyap, Tomasz Ksiezyk, Gale Martin, Marian H. Nodine, Mosfeq Rashid, Marek Rusinkiewicz, Ray Shea, C. Unnikrishnan, Amy Unruh, and Darrell Woelk. Infosleuth: Semantic integration of information in open and dynamic. In *SIGMOD Conference*, pages 195–206, Tuczon, AZ, 1997.

[8] T.C. Bell, A. Moffat, I.H. Witten, , and J. Zobel. The mg retrieval system: compressing for space and speed. *Communications of the Association for Computing Machinery*, 38(4):41–42, April 1995.

[9] M.A. Bickel. Automatic correction to misspelled names: a fourthgeneration language approach. *Communications of the ACM*, 30(3):224–228, 1987.

[10] D. Bitton and D. J. DeWitt. Duplicate record elimination in large data files. *ACM Transactions on Database Systems*, 8(2):255– 265, June 1983.

[11] K.D. Bollacker, S. Lawrence, and C.L. Giles. Citeseer: An autonomous web agent for automatic retrieval and identication of interesting publications. In *Proc. of 2nd Int. Conf. on Autonomous Agents*, Minneapolis, USA, 1998.

[12] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[13] S. Bressan, K. Fynn, C. H. Goh, M. Jakobisiak, K. Hussein, H. Kon, T. Lee, S. Madnick, T. Pena, J. Qu, A. Shum, and M. Siegel. The context interchange mediator prototype. In *Proc. ACM SIGMOD/PODS Joint Conference*, Tuczon, AZ, 1997.

[14] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. A principled approach to data integration and reconciliation in data warehousing. In *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)*, 1999.

[15] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Data integration in data warehousing. *Int. J. of Cooperative Information Systems*, 10:237–271, March 2001.

[16] A. Chatterjee and A. Segev. Data manipulation in heterogeneous databases. *SIGMOD Record*, 20(4):64–68, December 1991.

[17] K. W. Church and W. A. Gale. Probability scoring for spelling correction. *Statistics and Computing*, 1:93–103, 1991.

[18] William W. Cohen. Knowledge integration for structured information sources containing text. In *SIGIR-97 Workshop on Networked Information Retrieval*, Philidelphia, PA, 1997.

[19] William W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *SIGMOD Conference*, pages 201–212, Seattle, WA, 1998.

[20] William W. Cohen. A web-based information system that reasons with structured collections of text. In *Proceedings of the second internantional conference on Autonomous Agents*, Minneapolis, MN, 1998.

[21] William W. Cohen. The whirl approach to integration: An overview. In *AAAI-98 Workshop on AI and Information Integration*, Madison, WI, 1998.

[22] William W. Cohen. Whirl: A word-based information representation language. *submitted for journal publication*, 2000.

[23] William W. Cohen and Haym Hirsh. Joins that generalize: Text classification using whirl. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, New York, NY, 1998.

[24] Son Dao and Brad Perry. Applying a data miner to heterogeneous schema integration. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining(KDD-95)*, Menlo Park, CA, 1995.

[25] David J. DeWitt, Jeffrey F. Naughton, and Donovan A. Schneider. An evaluation of non-equijoin algorithms. In *VLDB*, pages 443–452, Barcelona, Spain, 1991.

[26] D. Fang, J. Hammer, and D. McLeod. The identification and resolution of semantic heterogeneity in multidatabase systems. In *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, pages 52–60. IEEE Computer Society Press, Los Alamitos, CA, 1994.

[27] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge discovery and data mining toward a unifying framework. In *Proceeding of The Second Int. Conference on Knowledge Discovery and Data Mining*, pages 82–88, Portland, OR, 1996.

[28] I. P. Fellegi and A. B. Sunter. A theory for record-linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[29] William Frakes and Ricardo Baeza-Yates. *Information retrieval: Data structures and algorithms*. Prentice Hall, 1992.

[30] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

[31] M. Ganesh. *Mining Entity-Identification Rules for Database Integration, Phd. Thesis.* Technical Report TR 97-041, Univ. of Minnesota, 1997.

[32] M. Ganesh, J. Sirvastava, and T. Richardson. Mining entity-identification rules for database integration. In *Proceedings of the Second International Conference on Data Mining and Knowledge Discovery*, pages 291–294, Portland, OR, 1996.

[33] Hector Garcia-Molina, Dallan Quass, Yannis Papakonstantinou, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, and Jennifer Widom. The

tsimmis approach to mediation: Data models and languages. In *NGITS (Next Generation Information Technologies and Systems)*, Naharia, Isreal, 1995.

[34] Michael R. Genesereth, Arthur M. Keller, and Oliver M. Duschka. Infomaster: An information integration system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, AZ, 1997.

[35] C. Lee Giles, Kurt Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *The Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, 1998.

[36] Henry G. Goldberg and Ted E. Senator. Restructuring databases for knowledge discovery by consolidation and link formation. In *Second International conference on Knowledge Discovery and Data Mining*, Portland, OR, 1996.

[37] J. T. Grudin. Error patterns in novice and skilled transcription typing. In W. E. Cooper, editor, *Cognitive Aspects of Skilled Typewriting*, pages 121–142. Springer-Verlag, 1983.

[38] J. Hammer, H. Garcia-Molina, J. Widom, W Labio, and Y. Zhuge. The stanford data warehousing project. *IEEE Data Engineering Bulletin*, 18(2):41–48, June 1995.

[39] Mauricio Hernandez and Salvatore J. Stolfo. *A generalization of band joins and the merge/purge problem.* Technical report CUCS005, Department of Computer Science, Columbia University, 1995.

[40] Mauricio Hernandez and Salvatore J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM-SIGMOD Conference*, San Jose, CA, 1995.

[41] Mauricio Hernandez and Salvatore J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. In *Data Mining and Knowledge Discovery*, pages 1–31, New York, NY, 1998.

[42] Timothy Huang and Stuart Russell. Object identification in bayesian context. In *Proceedings of IJCAI-97*, Nagoya, Japan, 1997.

[43] Timothy Huang and Stuart Russell. Object identification: A bayesian analysis with application to traffic surveillance. *Artificial Intelligence*, 103:1–17, 1998.

[44] S. B. Huffman and D. Steier. Heuristic joins to integrate structured heterogeneous data. In *1995 AAAI Spring Symposium on Information Gathering in Distributed, Heterogenous Environments*, Stanford, CA, 1995.

[45] S. B. Huffman and D. Steier. A navigation assistant for data source selection and integration. In *1995 AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, Cambridge, MA, 1995.

[46] Jemy A. Hylton. *Identifying and merging related bibliographic records. M.S. thesis*. MIT Laboratory for Computer Science Technical Report 678, 1996.

[47] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84:414–420, 1989.

[48] W. Kim, I. Choi, S. Gala, and M. Scheevel. On resolving schematic heterogeneity in multidatabase systems. *Distributed and Parallel Databases*, 1(3):251– 279, July 1993.

[49] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Pragnesh Jay Modi, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. Modeling web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998.

[50] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. The ariadne approach to web-based information integration. *International the Journal on Cooperative Information Systems (IJCIS), Special Issue on Intelligent Information Agents: Theory and Applications*, 10(1):145–169, 2001.

[51] K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, 1992.

[52] Steve Lawrence, Kurt Bollacker, and C. Lee Giles. Autonomous citation matching. In *Proceedings of the Third International Conference on Autonomous Agents*, New York, 1999.

[53] Mong Li Lee, Hongjun Lu, Tok Wang Ling, and Yee Teng Ko. Cleansing data for mining and warehousing. In *10th International Conference on Database and Expert Systems Applications (DEXA99)*, Florence, Italy, 1999.

[54] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Query answering algorithms for information agents. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, 1996.

[55] Wen-Syan Li. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the 20th VLDB Conference*, pages 1–12, Santiago, Chile, 1994.

[56] E-P. Lim, J. Srivastava, S. Prabhakar, and J. Richardson. *Entity identification in database integration.* Technical Report TR 92-62, Univ. of Minnesota, 1992.

[57] E-P. Lim, J. Srivastava, S. Prabhakar, and J. Richardson. Entity identification in database integration. In *Proceedings Ninth International Conference on Data Engineering, Vienna*, pages 294–301, Austria, 1993.

[58] E.-P. Lim, J. Srivastava, and S. Shekhar. An evidential reasoning approach to attribute value conflict resolution in database integration. *IEEE Transactions on Knowledge and Data Engineering*, 8(5):707–723, 1996.

[59] E.P. Lim, J. Srivastava, , and S. Shekhar. Resolving attribute incompatibility in database integration: An evidential reasoning approach. In *Proc. of 10th IEEE Data Eng. Conf.*, Houston, TX, 1994.

[60] H. Lu, W. Fan, C-H. Goh, S. Madnick, and D. Cheung. Discovering and reconciling semantic conflicts: A data mining prospective. In *IFIP Working Conference on Data Semantics (DS-7)*, Switzerland, 1997.

[61] H. Lu, B. Ooi, and C. Goh. Multidatabase query optimization: Issues and solutions. In *Proc. RIDE-IMS '93*, pages 137–143, 1993.

[62] Andrew McCallum, Kamal Nigam, and Lyle Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *In Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2000)*, 2000.

[63] Stephen McKearney and Huw Roberts. Reverse engineering databases for knowledge discovery. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.

[64] Alvaro Monge and Charles P. Elkan. The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, OR, 1996.

[65] Alvaro Monge and Charles P. Elkan. The webfind tool for finding scientific papers over the worldwide web. In *3rd International Congress on Computer Science Research*, Tijiuna, Mexico, 1996.

[66] Alvaro Monge and Charles P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *The proceedings of the SIGMOD 1997 workshop on Data Mining and Knowledge Discovery*, Tuczon, AZ, 1997.

[67] H. B. Newcombe and J. M. Kenedy. Record linkage. *Communications of the Association for Computing Machinery*, 5:563–566, 1962.

[68] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Learning to classify text from labeled and unlabeled documents. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[69] Daniel O'Leary. Internet-based information and retrieval systems. *Decision Support Systems*, 27(3):319–327, 1999.

[70] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithm and Complexity*. Prentice-Hall Inc, Englewood Cliffs, New Jersey, 1982.

[71] M. Perkowitz and O. Etzioni. Category translation: Learning to understand information on the internet. In *IJCAI*, pages 930–936, Montreal, Canada, 1995.

[72] Mike Perkowitz, Robert B. Doorenbos, Oren Etzoni, and Daniel S. Weld. Learning to understand information on the internet: An example-based approach. *Journal of Intelligent Information Systems*, 8(2):133–153, April 1997.

[73] Jose C. Pinheiro and Don X. Sun. Methods for linking and mining massive heterogeneous databases. In *Fourth International conference on Knowledge Discovery and Data Mining*, New York, NY, 1998.

[74] Jose C. Pinheiro and Don X. Sun. *Methods for linking and mining massive heterogeneous databases*. Technical memorandum, Bell Laboratories, Lucent Technologies, 1998.

[75] J. J. Pollock and A. Zamora. Automatic spelling correction in scientific and scholarly text. *ACM Computing Surveys*, 27(4):358–368, 1987.

[76] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[77] Calton Pu. Key equivalence in heterogenous databases. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, pages 314–316, 1991.

[78] J. R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[79] J.S. Ribeiro, K.A. Kaufman, and L. Kerschberg. Knowledge discovery from multiple databases. In *Knowlege Discovery and Datamining*, Menlo Park, CA, 1995.

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

[81] T. Senator, H. Goldberg, J. Wooton, A. Cottini, A. Umar, C. Klinger, W. Llamas, M. Marrone, , and R. Wong. The fincen artificial intelligence system: Identifying potential money laundering from reports of large cash transactions. In *Proceedings of the 7th Conference on Innovative Applications of AI*, Montreal, Canada, 1995.

[82] H. S. Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Computational Learing Theory*, pages 287–294, 1992.

[83] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Surveys*, 22:183–236, September 1990.

[84] E. Simoudis, B. Livezey, and R. Kerber. Using recon for data cleaning. In *Knowledge Discovery and Datamining*, pages 282 – 287, Menlo Park, CA, 1995.

[85] Y.R. Wang and S.E. Madnick. The inter-database instance identification problem in integrating autonomous systems. In *Proc. of the Int'l Conf. on Data Eng.*, Los Angeles, CA, 1989.

[86] Gio Wiederhold. Intelligent integration of information. In *Proceedings of ACM SIGMOD conference on manangement of data*, pages 434–437, Washington, DC, May 1993.

[87] William Winkler. *Record Linkage Software and Methods for Merging Administrative Lists*. Statistical research division Technical Report RR01—03, U.S. Bureau of Census, 2001.

[88] William E. Winkler. Advanced mathods for record linkage. In *Proceedings of the Section of Survey Research Methods*, pages 467–472, American Statistical Association, 1994.

[89] William E. Winkler. The state of record linkage and current research problems. In *Proceedings of the section of survey methods*, Canada, 1999.

[90] Tak W. Yan and Hector Garcia-Molina. Duplicate removal in information dissemination. In *Proceedings of VLDB*, Zurich, Switzerland, 1995.

[91] G. Zhou, R. Hull, R. King, and J-C. Franchitti. Using object matching and materialization to integrate heterogeneous databases. In *Proc. of Third Intl. Conf. On Cooperative Information Systems (CoopIS-95)*, Vienna, Austria, 1995.