Pergamon

0306-4379(01)00042-4

Information Systems Vol. 26, No. 8, pp. 607–633, 2001
Copyright© 2001 Elsevier Science Ltd
All rights reserved
0306-4379/01

LEARNING OBJECT IDENTIFICATION RULES FOR INFORMATION INTEGRATION

Sheila Tejada[1], Craig A. Knoblock[1], and Steven Minton[2]

[1]University of Southern California/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey CA 90292
[2]Fetch Technologies 4676 Admiralty Way, Marina del Rey CA 90292

(December 2001)

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. This manual process is time consuming and error-prone. In our approach, Active Atlas learns to tailor mapping rules, through limited user input, to a specific application domain. The experimental results demonstrate that we achieve higher accuracy and require less user involvement than previous methods across various application domains.

1. INTRODUCTION

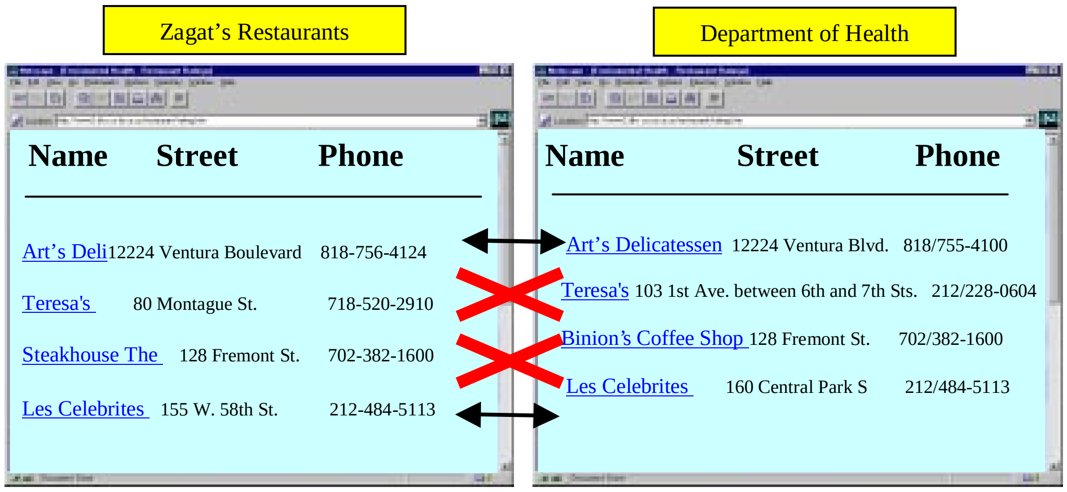
Many problems arise when integrating information from multiple information sources on the web [79]. One of these problems is that data objects can exist in inconsistent text formats across several sources. An example application of information integration involves integrating all the reviews of restaurants from the Zagat's Restaurants webpage 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. In the first example the restaurant referred to as "Art's Deli" on the Zagat's webpage 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 determine if the objects are the same. When two objects are determined to be the same, a *mapping* is created between them.

The examples in Figure 1 are each representative of a type of example 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. 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 errors 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 attribute information is captured in the form of object identification rules (*mapping rules*), which are used to determine the mappings between the objects.

This paper presents an object identification system called Active Atlas that learns to tailor mapping rules 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



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

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 St.	702/382-1600
Les Celebrites	160 Central Park S	212/484-5113

Fig. 1: Matching Restaurant Objects

the highest possible accuracy in object identification with minimal user interaction in order to properly integrate data from multiple information sources. Active Atlas is a tool that can be used in conjunction with information mediators, such as SIMS [4] and Ariadne [47], to properly handle the object identification problem for information integration.

1.1. Ariadne Information Mediator

The Ariadne information mediator [48] 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 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.

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 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 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 4) is used to describe the contents of the individual sources. Given a query

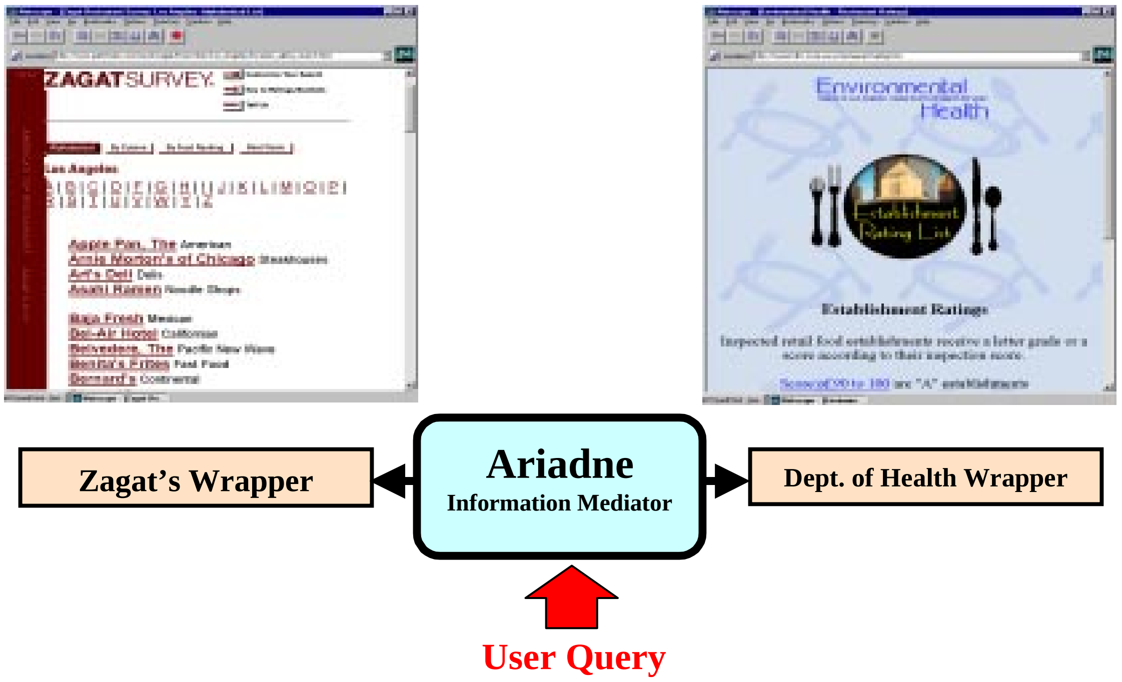
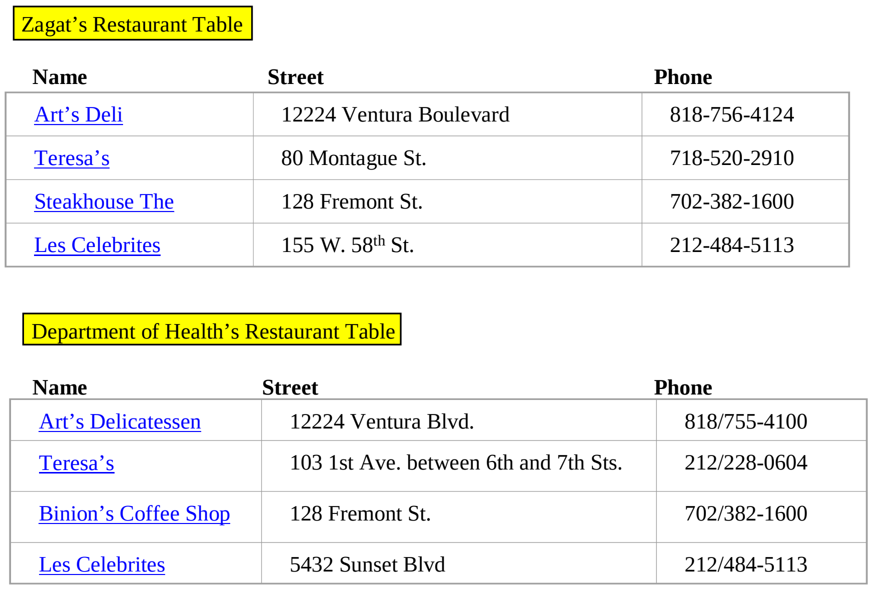


Fig. 2: Restaurant Application



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 St.	702/382-1600
Les Celebrites	5432 Sunset Blvd	212/484-5113

Fig. 3: Restaurant data objects stored as records

in terms of the concepts (e.g. Restaurant) and attributes (e.g. Name and Phone) described in the model, the system dynamically selects an appropriate set of sources and then generates a plan to efficiently produce the requested data.

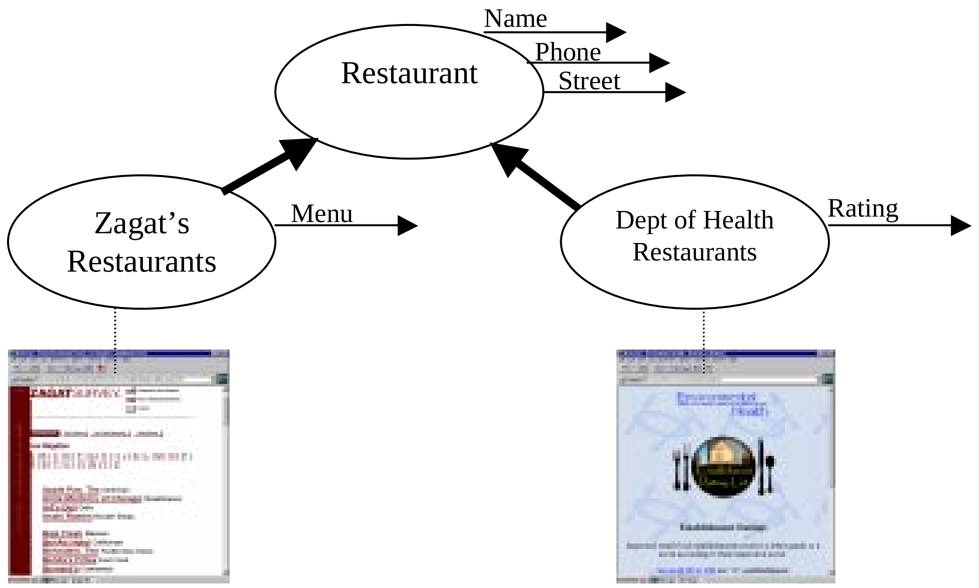


Fig. 4: Restaurant Domain Model

Unfortunately, due to the object identification problem described in Figure 1, the given domain model (Figure 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 5).

After Active Atlas determines the total mapping assignment for an application, it builds two 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.

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 Restaurants **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

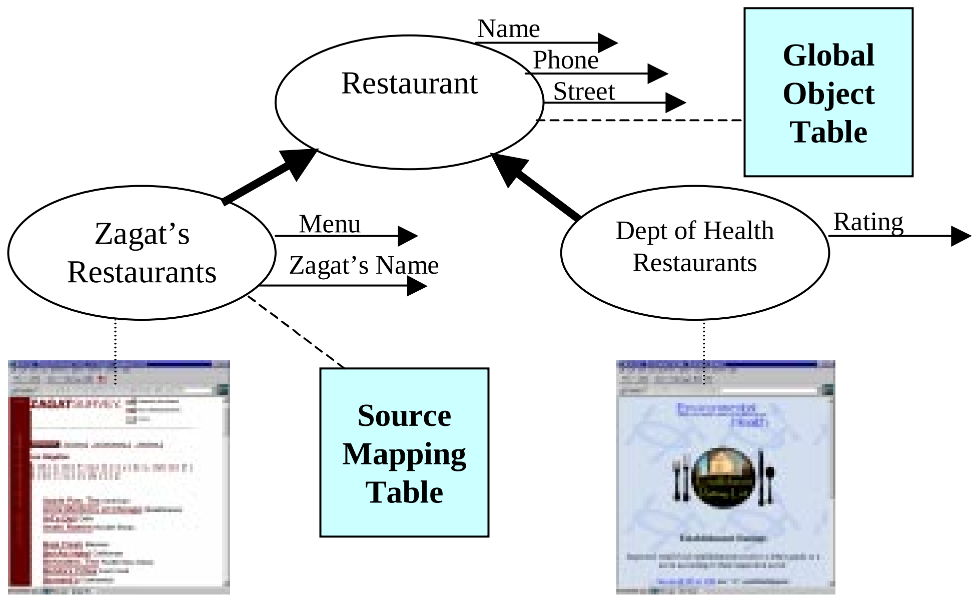


Fig. 5: Restaurant Domain Model with Mapping Table

same. Once these mapping constructs, i.e. mapping tables or functions, are have been automatically generated, they can be considered new information sources (Figure 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.

1.2. Approach

Identifying mappings between objects may be dependent on the application. The information necessary for deciding a mapping may not be evident by solely evaluating the data itself because there might 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 6) 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.

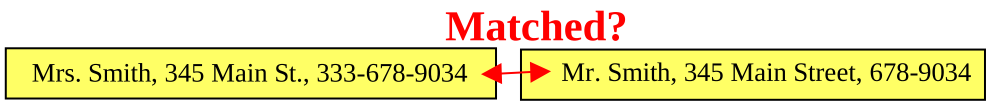

Matched?
Mrs. Smith, 345 Main St., 333-678-9034
Mr. Smith, 345 Main Street, 678-9034

Fig. 6: Example Census Data

In the restaurant domain (Figure 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, 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, for example, telemarketers trying to sell long distance phone service to these restaurants, would only be interested in the restaurant phone number matching - the other attributes would be irrelevant. Because the mapping assignment can depend on the application, the data itself is not enough to decide the mappings between the sets of objects. The user's knowledge 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. Our approach is to learn a set of mapping rules for a specific application domain, through limited user input. We accomplish this by first determining the text formatting transformations, and then learning domain-specific mapping rules.

The focus of this paper is to present a general method of learning mapping rules for object identification. This paper consists of five sections. Section 2 provides a detailed description of our object identification system Active Atlas. Section 3 presents the experimental results of the system. Section 4 discusses related work, and Section 5 concludes with a description of future work.

2. LEARNING OBJECT IDENTIFICATION RULES

We have developed an object identification method that learns the mapping information necessary to properly integrate web sources with high accuracy. As shown in Figure 7, there are two stages in our method, computing the similarity scores and mapping-rule learning. In the first stage the candidate generator is used to propose the set of possible mappings between the two sets of objects by comparing the attribute values and computing the similarity scores for the proposed mappings. In the next stage the mapping rule learner determines which of the proposed mappings are correct by learning the appropriate mapping rules for the application domain.

The objects from the sources are given as input in the first stage of processing. The candidate generator compares all of the shared attributes of the given objects by applying a set of domain-independent functions to resolve text transformations, e.g., Substring, Acronym, Abbreviation, etc., to determine which objects are possibly mapped together. It then computes the similarity scores for each of attributes of the proposed mapped pairs of objects or *candidate mappings*. The output of the candidate generator is the set of candidate mappings, each with their corresponding set of attribute similarity scores and combined total score.

The second stage (Figure 7) is learning the mapping rules. This component determines 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 our 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.

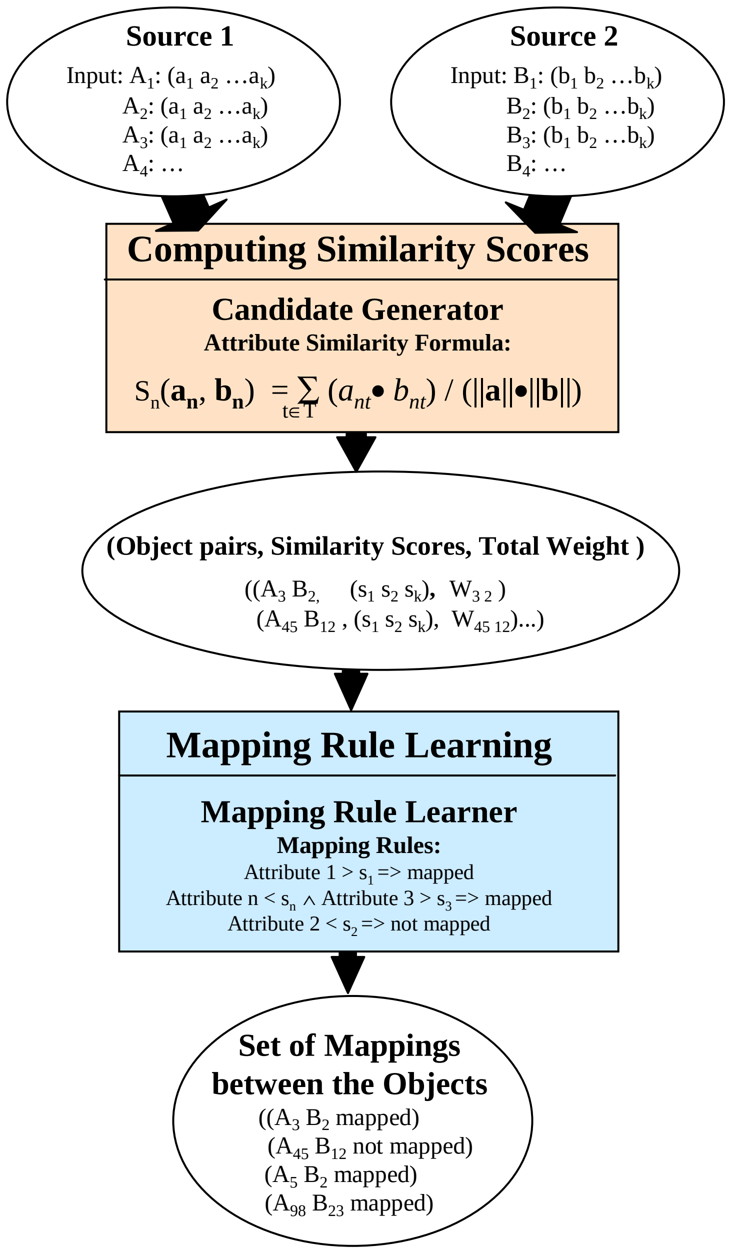

Source 1
Input: A_1: (a_1 a_2 …a_k)
A_2: (a_1 a_2 …a_k)
A_3: (a_1 a_2 …a_k)
A_4: …
Source 2
Input: B_1: (b_1 b_2 …b_k)
B_2: (b_1 b_2 …b_k)
B_3: (b_1 b_2 …b_k)
B_4: …
Computing Similarity Scores
Candidate Generator
Attribute Similarity Formula:
$S_n(\mathbf{a_n}, \mathbf{b_n}) = \sum_{t \in T} (a_{nt} \bullet b_{nt}) / (\|\mathbf{a}\| \bullet \|\mathbf{b}\|)$
(Object pairs, Similarity Scores, Total Weight)
((A_3 B_2, (s_1 s_2 s_k), $W_{3\,2}$)
(A_{45} B_{12} , (s_1 s_2 s_k), $W_{45\,12}$)...)
Mapping Rule Learning
Mapping Rule Learner
Mapping Rules:
Attribute 1 > s_1 => mapped
Attribute n < s_n ∧ Attribute 3 > s_3 => mapped
Attribute 2 < s_2 => not mapped
Set of Mappings between the Objects
((A_3 B_2 mapped)
(A_{45} B_{12} not mapped)
(A_5 B_2 mapped)
(A_{98} B_{23} mapped)

Fig. 7: General System Architecture

2.1. Computing Similarity Scores

When comparing objects, the alignment of the attributes is determined by the domain model (Figure 4). The values for each attribute are compared individually (Figure 8 – **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 decreases mapping error by reducing the number of candidate mappings considered.

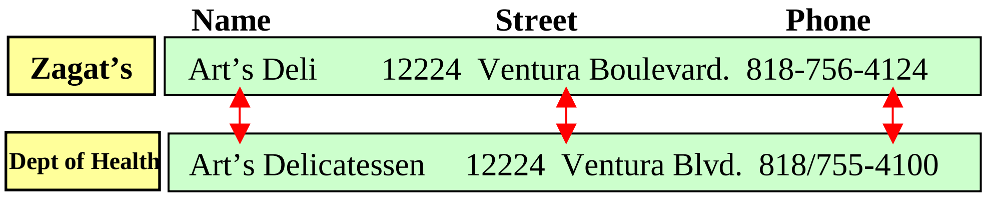

Name
Street
Phone
Zagat's
Art's Deli 12224 Ventura Boulevard. 818-756-4124
Dept of Health
Art's Delicatessen 12224 Ventura Blvd. 818/755-4100

Fig. 8: Comparing Objects by Attributes

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 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. The process for computing these scores is described in detail later in this section.

Example attribute similarity scores for the candidate mapping of the "Art's Deli" and "Art's Delicatessen" objects are displayed in Figure 9. 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.

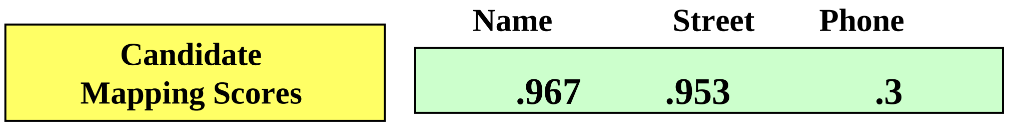


Fig. 9: Set of Computed Similarity Scores

2.1.1. General Transformation Functions

Included in our framework is a set of general domain-independent transformation functions to resolve the different text formats used by the objects (Figure 10). These transformation functions (e.g. Abbreviation, Acronym, Substring, etc.) are domain-independent and are applied to all of the attribute values in every application domain. These functions determine if text transformations exist between words *(tokens)* in the attribute values, e.g., (Abbreviation - "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.

There are two basic types of the transformation functions. Type I transformations require only a single token as input in order to compute its transformation. Type II transformations compare tokens from two objects.

Type I transformations

- **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** replaces token with corresponding abbreviation (e.g., 3rd or third).

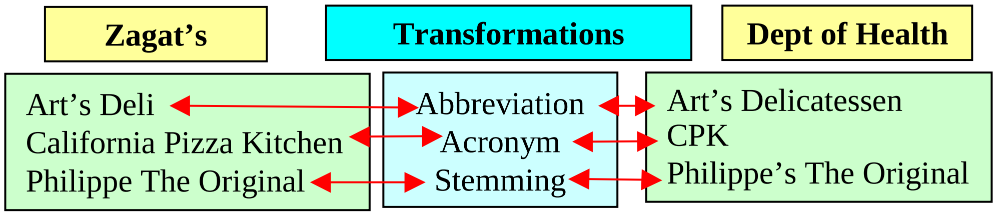


Fig. 10: Transformations

Type II transformations

- **Equality** compares two tokens to determine if each token contains the same characters in the same order.
- **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.
- **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).

2.1.2. Information Retrieval Engine

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

Most information retrieval systems, such as the Whirl system [19], apply only the stemming transformation function to compare the words of the attribute values. The stemming transformation function compares the stem or root of the words to determine similarity. In our approach we have included a set of general transformation functions, 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. These tokens are compared using the transformation functions. 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's", "Art's"), i.e. exact text match, and (Abbreviation – "Deli", "Delicatessen") (Figure 11).

The information retrieval (IR) engine has been modified in order to apply the transformations in a three step process. The first step is to apply all of the Type I transformations to the data. The second step after applying the Type I transformations is to determine the set of object pairs

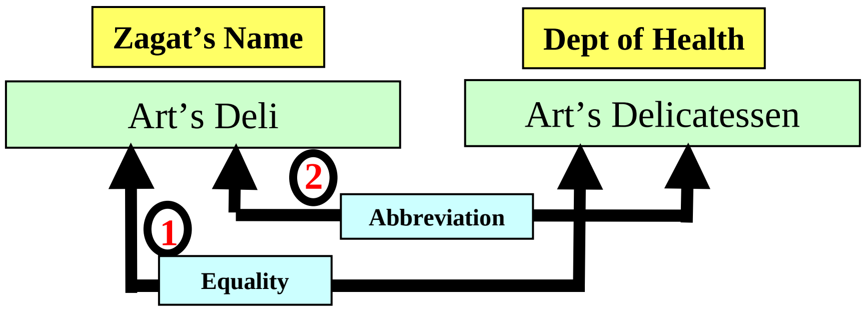


Fig. 11: Applying Transformation Functions

that are related. And the third step is to apply the Type II transformations in order to further strengthen the relationship between the two data objects.

In step 1, the Type I transformations are applied to one attribute dataset, such as the Department of Health's restaurant names. Like a traditional IR engine the data is first tokenized, the transformations are applied to each token (Stemming, Soundex, etc.) and then stored in a hash-table. Therefore, each token can have more than one entry in the hash-table. In this modified IR engine, information about which transformation was applied is stored, as well as the token frequency and tuple (object) number.

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 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 hash-table. The hash-table entries returned contain the tuple numbers of the related objects from the other dataset (Department of Health). This process is completed for every attribute (**Name**, **Street**, and **Phone**).

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. Once this is done, then the more computationally expensive transformations (Type II) 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 relate them, along with the token frequency information needed for computing the similarity scores.

2.1.3. Computing Attribute Similarity Scores

Each transformation function has an initial probability score or transformation weight that is used in the calculation of the attribute similarity scores. For the example in Figure 11 the total similarity score for the **Name** attribute values "Art's Deli" and "Art's Delicatessen," is calculated with the weights for the two transformations (Equality – "Art's" "Art's"), and (Abbreviation – "Deli", "Delicatessen"). In this section we discuss in detail how the transformation weights are used to calculate the attribute similarity scores.

Figure 12 shows how attribute values, (**Z1, Z2, Z3**) and (**D1, D2, D3**), are compared in order to generate the set of attribute similarity scores ($\mathbf{S_{name}}$, $\mathbf{S_{street}}$, $\mathbf{S_{phone}}$). In Figure 12, **Z1** and **D1** represent **Name** attribute values, e.g. "Art's Deli" and "Art's Delicatessen." These values are compared using general transformation functions and then their transformation weights are used to calculate the similarity score $\mathbf{S_{name}}$.

To calculate the similarity scores for the attribute values of the candidate mappings, we have employed the cosine measure commonly used in information retrieval engines with an altered form of the TFIDF (Term Frequency x Inverse Document Frequency) weighting scheme [28]. Because the attribute values of the object are very short, the term frequency weighting is not relevant.

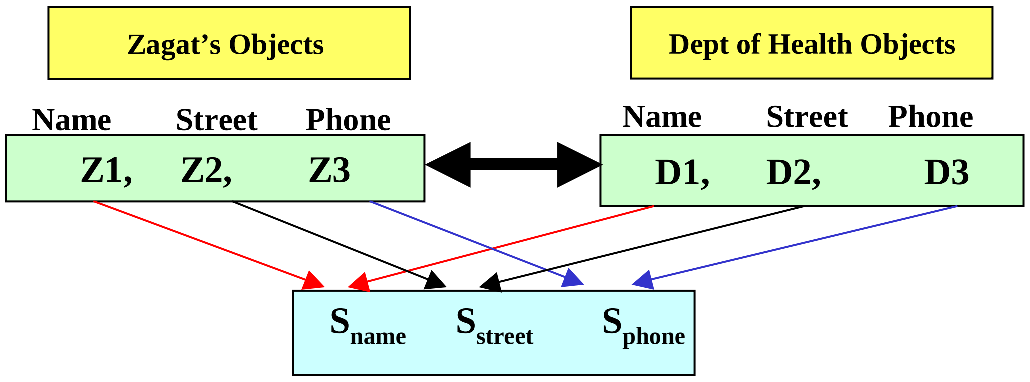


Fig. 12: Computing Similarity Scores

In place of the term frequency, we have substituted the initial probability score of the applied transformation function, which represents the transformation weight (TWIDF). The similarity score for a pair of attribute values is computed using the following attribute similarity formula:

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

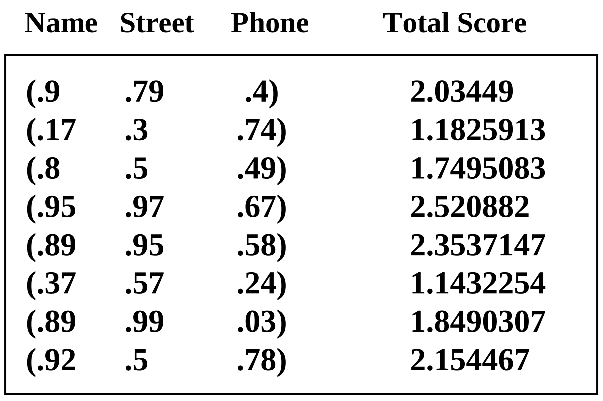
- $\mathrm{w_{ia}}$= (0.5 + 0.5 $\mathrm{tweight_{ia}}$/max $\mathrm{tweight_{a}}$) x IDF
- $\mathrm{w_{ib}}$= *freq*$_{\mathrm{ib}}$ x $\mathrm{IDF_i}$
- $\mathrm{tweight_{ia}}$ = transformation function weight *i* for attribute value **a**
- max $\mathrm{tweight_a}$ = maximum transformation function weight
- $\mathrm{IDF_i}$= IDF (Inverse Document Frequency) of token *i* in the entire collection
- $\mathrm{freq_{ib}}$ = 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 number of tokens in the documents. The terms $\mathrm{w_{ia}}$ and $\mathrm{w_{ib}}$ correspond to the weights computed by the TWIDF weighting function. This formula measures the similarity by computing the distance between two attribute values.

2.1.4. Calculating the 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 Figure 13. 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 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. Once the total object scores are computed, the candidate mapping information is then given as input to the mapping-rule learner (Figure 13).



Name	Street	Phone	Total Score
(.9	.79	.4)	2.03449
(.17	.3	.74)	1.1825913
(.8	.5	.49)	1.7495083
(.95	.97	.67)	2.520882
(.89	.95	.58)	2.3537147
(.37	.57	.24)	1.1432254
(.89	.99	.03)	1.8490307
(.92	.5	.78)	2.154467

Fig. 13: Output of the Candidate Generator

2.2. Mapping-Rule Learning

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 our approach, 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 to learn high accuracy mapping rules.

2.2.1. 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* [72]. Decision tree learning is an inductive learning technique, where a learner constructs a decision tree (Figure 14) 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 14. 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 which are the most useful attributes to classify the examples. A metric called *information gain* measures how well an attribute divides the given set of training examples by their classification (mapped/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.

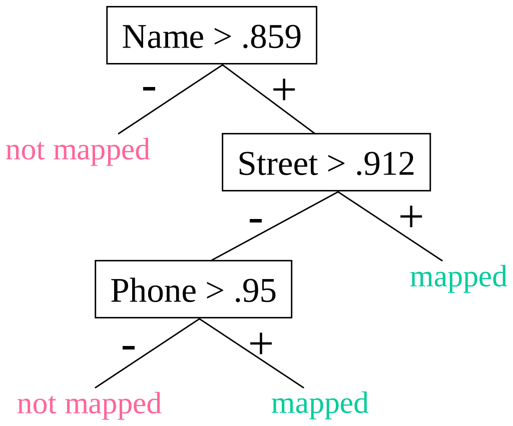


Fig. 14: Example decision tree for restaurant domain

2.2.2. Active Learning

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 section. To efficiently learn the mapping rules for a particular task or domain, we are currently applying a supervised learning technique, which uses a combination of several decision tree learners, called query by bagging [1]. 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. The query by bagging technique is considered an active learning technique [5] because the system actively chooses examples for the user to label. We have adopted this committee-based approach in order to reduce the number of user-labeled examples.

Figure 15 graphically shows the learning algorithm for the query by bagging technique. The first step is selecting the small initial set of training examples. The set of candidate mappings (Figure 13) 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 grouped according to their total similarity scores (Figure 13) – roughly corresponding to dividing up the examples by the number of attributes they highly match on, e.g. if there are three attributes being compared then there are three groups of examples. The group that matches on the most attributes tends to have the fewest examples and the majority of positive examples for the application domains. An equal number of examples are chosen randomly from each group to compose the initial training set. This selection process was developed to insure that a variety of examples would be included because there is 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.

Once the initial training set has been created, the next step is to use the *bagging* [12] 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 13). If the similarity scores of a candidate mapping fulfill the conditions of a rule then the candidate

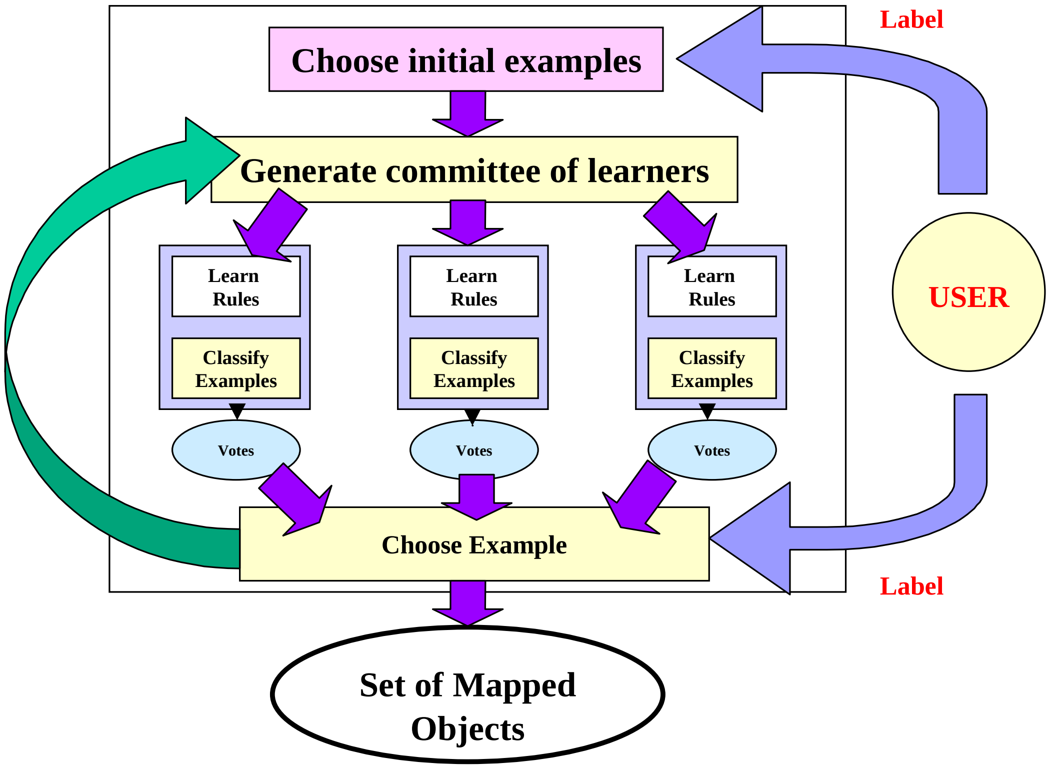
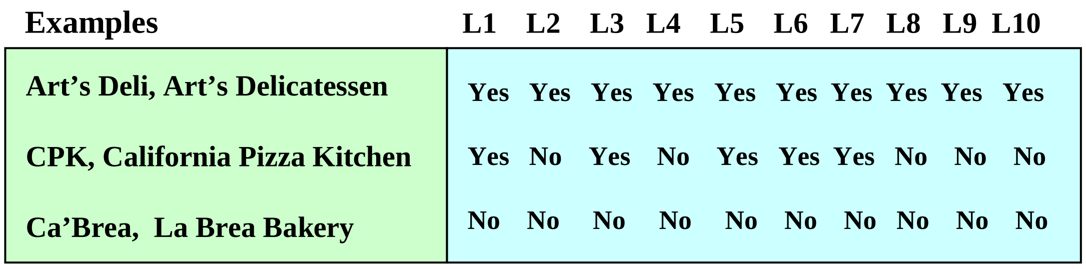


Fig. 15: Mapping-Rule Learner

mapping is classified as mapped.

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. The choice of an example is based on the disagreement of the query committee on its classification (Figure 16). The maximal disagreement occurs when there is 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 16 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

Fig. 16: Choosing the next example

There are cases where the committee disagrees the most on several examples. In order to break a tie and choose an example, the total object similarity score, associated with each example (Figure 13), is used as the deciding factor. Since there is 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. Therefore, the example with the greatest total object similarity score is chosen from the proposed set.

Once an example is chosen, the user is asked to label the example, and the system updates the committee. 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. After the total mapping assignment is complete, new information sources (mapping tables) can be created for use by an information mediator.

3. 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. For each domain, we also 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.

We also include results from two baseline experiments as well. The first baseline experiment runs the candidate generator to compare all of the shared attributes of the objects separately, while the second baseline experiment compares the objects as a whole, with all of the shared attributes concatenated into one attribute. Both baseline experiments require choosing an optimal mapping threshold from a ranked list of candidate mappings, and they use only the stemming transformation function.

3.1. 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. 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.

3.1.1. 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 baseline experiments, 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 divided by the total number of mappings plus the number of correct candidate mappings not proposed. This is comparable to the Whirl system [19].

Listed in the following table is the accuracy information at specific thresholds for the baseline experiment that compared the attributes separately (Figure 17). The mapping threshold with the highest accuracy is highlighted. In Figure 17 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 cannot be chosen to achieve perfect accuracy. This is true as well for the second baseline experiment which compares the object as a whole. For that experiment, at the optimal threshold there are also 109 correct mappings, 3 true mappings and 10 false mappings. 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	0.9871
92	91	21	1	0.9931
111	109	3	2	0.9980
116	110	2	6	0.9977
119	111	1	8	0.9972
128	112	0	16	0.9951

Fig. 17: Baseline Results (separate attributes using stemming)

We compared the accuracy for the best cases of the baseline results against the accuracy of the two systems for mapping-rule learning. The purpose of the Passive Atlas experiments are to show that learning the mapping rules can achieve higher accuracy than the baseline experiments, 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 18.

The accuracy results from the four types of experiments are shown in relation to the number of examples that were labeled. For the results of the baseline experiments, only the optimal accuracy is displayed, because the user chooses a threshold and does not label examples. In this domain the baseline experiments demonstrate that comparing the attributes individually does improve the accuracy of the mappings. When comparing the objects as a whole, words can overlap between the attribute values. This increases mapping error by increasing the number of false candidate mappings considered.

For the two learning systems the results have been averaged over 10 runs, and the learners classified 3259 candidate mappings proposed by the candidate generator. In these experiments, the initial probability scores for all of the general transformation functions are assigned by the user, based on background knowledge of the transformation functions. The initial probability scores for the transformation functions are only set once. They remain constant for all the experiments across all three domains.

Figure 18 shows that learning the mapping rules increases the accuracy of the mapping assignment. Active Atlas exceeds the highest possible accuracy for the baseline case comparing separate attributes at 60 examples (at 900 examples for Passive Atlas). In the Active Atlas experiments, the system achieved 100% accuracy at 80 examples, while Passive Atlas reached 100% accuracy at 1900 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 random manner, independent of the actual data. Because these domains have a sparse number of positive (mapped) examples (Restaurant 3%, Companies

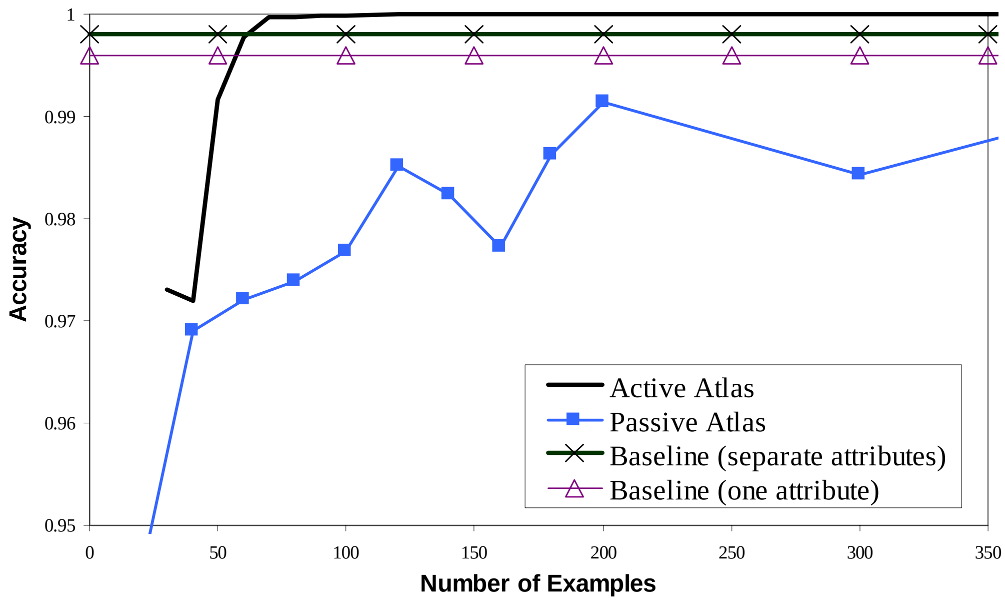


Fig. 18: Restaurant domain experimental results

3%, and Airports 9%), 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.

We also conducted two experiments (Figure 19) in order to examine the effect of the set of transformations on the accuracy of Active Atlas across the three domains. In the first experiment we ran Active Atlas with all of the transformation functions assigned to the same probability score of .5. We did this to test the sensitivity of the system to the setting of the initial probability scores. In the second experiment we ran a variation of Active Atlas where it applies only the stemming transformation function.

The results for the sensitivity analysis show that the initial probability scores do effect the performance of the system. This variation of Active Atlas achieves 100% accuracy at 250 examples for the Restaurant domain. The system's performance for the Company and Airport domains decreased but not as significantly. The sensitivity analysis also demonstrates that all transformation functions should not have the same weight. For example, in the sensitivity experiment the transformation (Abbreviation - "Deli", "Delicatessen") has the same weight as the transformation (Substring - "cat", "Delicatessen"). The transformation (Abbreviation - "Deli", "Delicatessen") accurately reflects a relationship that exists in this domain, and should have a greater weight than the other transformation. Our future work (Section 5) is concerned with learning to optimize the transformation function settings to achieve high accuracy using fewer labeled examples.

In the Active Atlas experiments shown in Figure 19, the system with the transformations achieved 100% accuracy at 80 examples, and the system with stemming achieved 100% accuracy at 140 examples. In this domain the transformations increase the similarity scores of a majority of candidate mappings. This means that the experiments that used the general transformations needed more examples initially, but then were able to find the harder examples more easily.

In the restaurant domain there are fewer text inconsistencies than the other two domains, and therefore, the transformation functions are not as critical for determining a mapping. This is the reason that Active Atlas (stemming) outperforms the transformation sensitivity experiment. In the other domains, where transformation functions are necessary, the transformation sensitivity experiment outperforms Active Atlas (stemming).

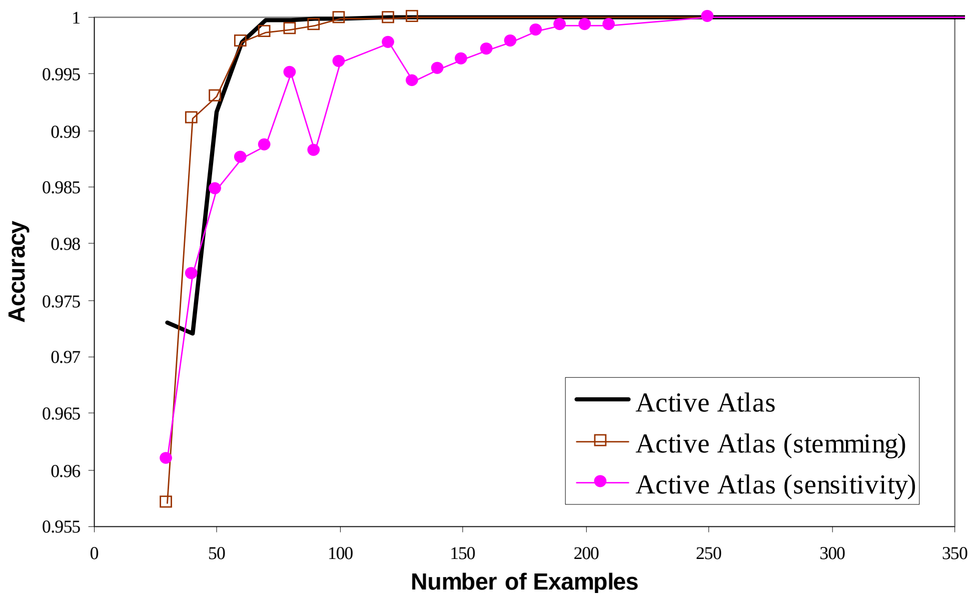


Fig. 19: Restaurant domain experimental results

3.2. Company Domain

In the company domain there are two websites, HooversWeb and IonTech, which both provide information on companies (**Name, Url** and **Description**). In this domain the **Url** attribute is usually a very good indicator for companies to match on, e.g. "Soundworks" (Figure 20). 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.

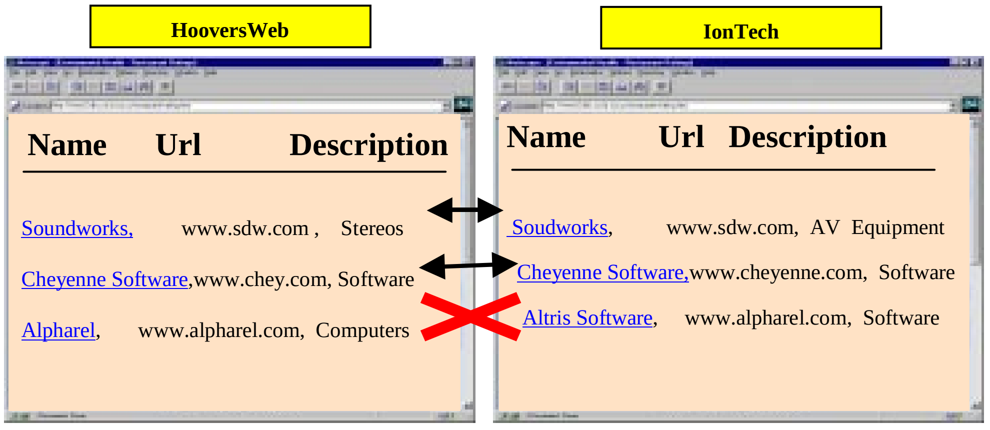

HooversWeb
IonTech
Name Url Description
Soundworks, www.sdw.com , Stereos
Cheyenne Software,www.chey.com, Software
Alpharel, www.alpharel.com, Computers
Name Url Description
Soudworks, www.sdw.com, AV Equipment
Cheyenne Software,www.cheyenne.com, Software
Altris Software, www.alpharel.com, Software

Fig. 20: Company Domain Examples

3.2.1. Experimental Results

In this domain HooversWeb has 1163 objects and the IonTech site has 957 objects. There are 315 correct mappings between the sites. The results for the baseline case comparing separate attributes are shown in Figure 21. The optimal threshold is at rank 282, where 41 examples are incorrectly classified and 12% of the object mappings are missing. For the second baseline experiment comparing objects as a whole, there are 119 incorrectly classified examples.

Ranked Examples	Correct Mappings	Missed Mappings	False Mappings	Accuracy
50	50	265	0	0.9723
223	222	93	1	0.9901
282	278	37	4	0.9957
390	308	7	82	0.9907
4919	315	0	4604	0.5189

Fig. 21: Baseline Results (separate attributes using stemming)

For the learning experiments (Figure 22) in the company domain, the candidate generator proposed 9570 candidate mappings, and the results have been averaged over 10 runs. Similar to the restaurant domain, Figure 22 shows that learning the mapping rules increases the accuracy of the mapping assignment. Active Atlas achieves higher accuracy than the baseline experiment with separate attributes at 80 examples (at 800 examples for Passive Atlas). The graph also shows that the active learner requires fewer labeled examples than Passive Atlas.

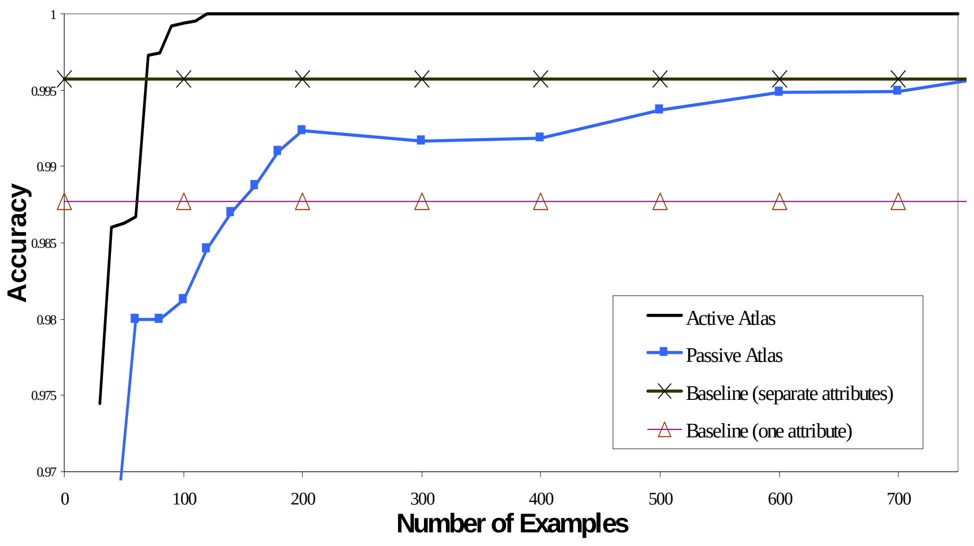


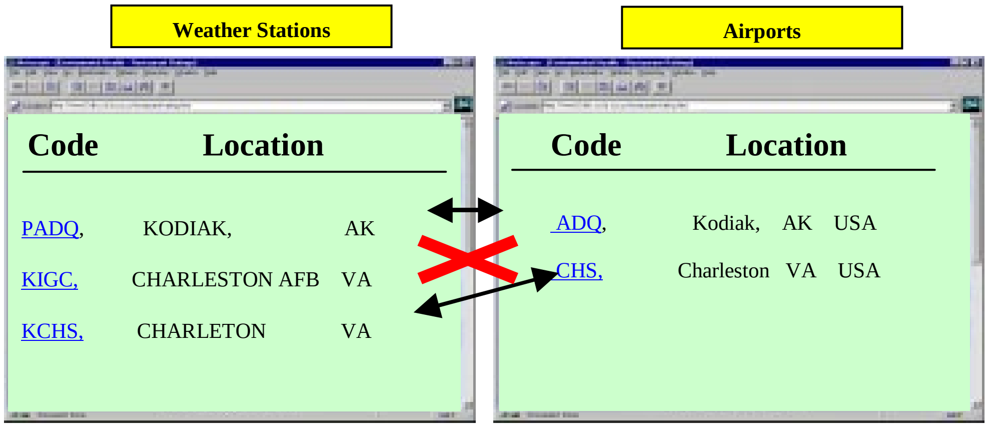
Fig. 22: Company Domain experimental results

The transformation functions have more influence in this domain. The transformation functions are able to resolve the spelling mistake between “Soundworks” and “Soudworks” (Figure 20) using the Soundex transformation function, which made the difference in it being mapped or unmapped.

In the Active Atlas experiments, the system with the transformations achieved 100% accuracy at 120 examples, the system with stemming achieved 100% accuracy at 900 examples, and the system for the sensitivity experiment achieved 100% at 180 examples.

3.3. *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 23.



Weather Stations

Code	Location	
PADQ,	KODIAK,	AK
KIGC,	CHARLESTON AFB	VA
KCHS,	CHARLETON	VA

Airports

Code	Location		
ADQ,	Kodiak,	AK	USA
CHS,	Charleston	VA	USA

Fig. 23: Airport/Weather Domain examples

3.3.1. *Experimental Results*

The results in this domain for the candidate generator at selected thresholds are shown in Figure 24. The optimal threshold is set at rank 438, where 229 examples are incorrectly classified and over 25% of the object mappings are missing. The accuracy for the baseline experiment is also dramatically decreased in this domain, with 309 incorrectly classified examples at the optimal threshold.

Ranked Examples	Correct Mappings	Missed Mappings	False Mappings	Accuracy
19	19	408	0	0.9052
355	276	151	79	0.9465
438	318	109	120	0.9468
479	331	96	148	0.9433
1667	400	27	1267	0.6996

Fig. 24: Baseline Results (separate attributes using stemming)

In this domain the set of transformation functions plays a larger role in increasing the accuracy of the object mappings, as clearly shown by the candidate generator results. The main reason for the lower accuracy of the experiments with stemming is because the baseline system is not able to recognize that the airport code is a substring of the weather code for the **Code** attribute. Therefore, it only uses the **Location** attribute to match objects, so it makes mistakes, such as mapping the "KIGC" and "CHS" objects in Figure 25. There are 27 object mappings that were not proposed by the candidate generator because it did not have the necessary transformations.

Figure 25 shows results from a baseline experiment comparing separate attributes and using the set of transformation functions. The significant effect of the transformation functions are clearly demonstrated in this domain. The optimal threshold is set at rank 368, where 76 examples are incorrectly classified. This is less than one third of the incorrect classifications made by the baseline experiment that compares separate attributes using stemming.

Ranked Examples	Correct Mappings	Missed Mappings	False Mappings	Accuracy
342	342	85	0	0.9801
351	350	77	1	0.9818
368	360	67	8	0.9825
648	407	20	241	0.9396
3282	420	7	2862	0.3297

Fig. 25: Baseline Results (separate attributes using set of transformations)

Like the other domains, Figure 26 shows that learning the mapping rules increases the accuracy of the mapping assignment. Active Atlas immediately achieves higher accuracy than the baseline experiments, and Passive Atlas achieves higher accuracy after 60 examples.

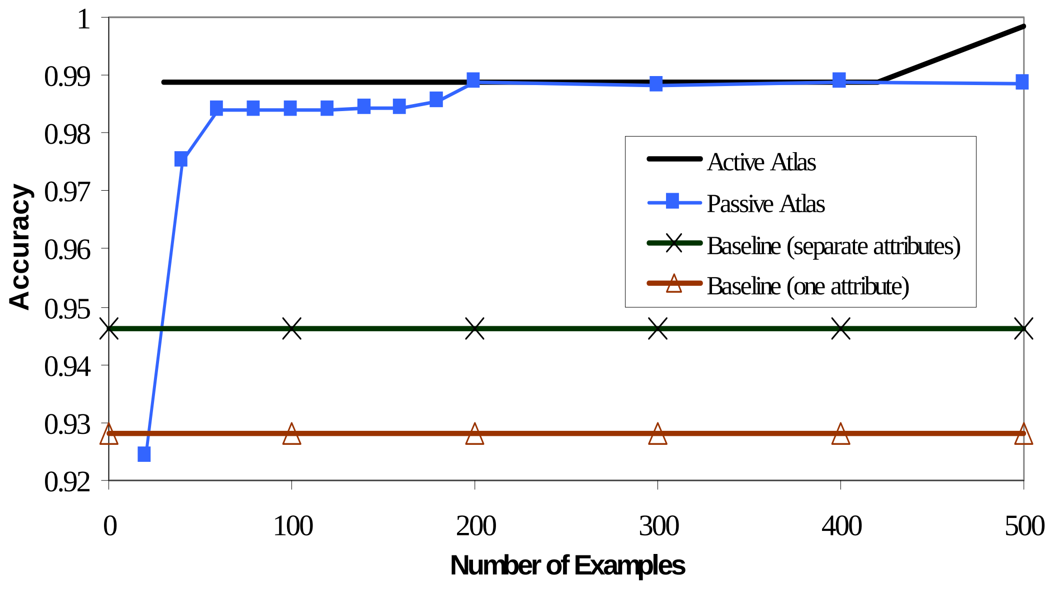


Fig. 26: Airport/Weather domain experimental results

The graph also shows that there are cases where (at 200 and 400 examples) Passive and Active Atlas achieve the same accuracy. This is because for Active Atlas, the committee of learners converged too quickly on a classification of the data; or in other words, all of their votes were identical. Therefore, many more labeled examples were needed to achieve high accuracy. As discussed in the Section 5, one of the goals for future work is to address this issue of reducing the number of labeled examples needed by Active Atlas to achieve a high accuracy mapping assignment.

4. 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. The main advantage of our system is that it can, with high accuracy, learn to tailor mapping rules 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 three application areas that included the problem of object identification (Figure 27): databases, information retrieval, and probabilistic methods.

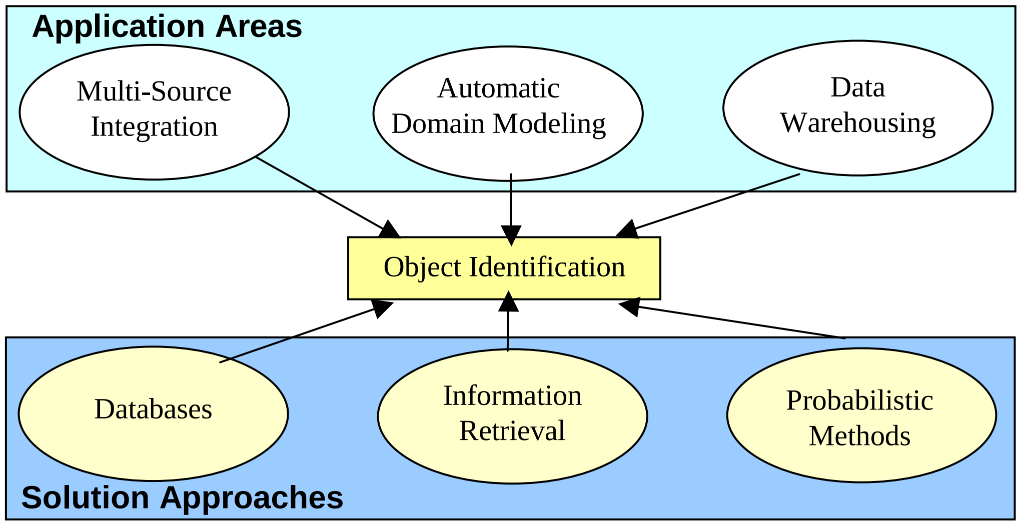


Fig. 27: Related work graph

4.1. Application Areas

Research in multi-source integration [79] 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, 34, 75]. General information integration systems, like Pegasus [2], TSIMMIS [31], Infomaster [32], InfoSleuth [7], COIN [13], and Information Manifold [51], 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 [66, 65, 82, 46, 25] 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 [23, 24, 52, 59, 73] 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 [26].

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 [36, 76, 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 [77]. Object identification techniques can be applied here to assist in cleaning the data [30].

4.2. Solution Approaches

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

4.2.1. 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, 35, 49, 69] have been applied by many object identification systems to measure text similarity [30, 29, 39, 38, 37, 44, 81]. 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 [62, 61, 60, 67, 68] that have used 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 [39, 38, 37], Lu et al [50, 57, 58], Chatterjee and Segev [16], and Pu [71] 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 [67, 68] and Ganesh et al [30, 29] 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 [46, 56, 54, 53, 55, 78, 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.

4.2.2. Information Retrieval

The problem of object identification has also appeared in the information retrieval community [74, 64]. 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, 22] 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 [70] is the only transformation used to calculate similarity between strings; therefore, in Figure 10 "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 [43, 42] 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, 33] is an information retrieval system that finds relevant and similar papers and articles, where similarity is based on the set of citations listed in the articles. Citeseer also uses domain-specific normalization techniques to standardize article citations.

4.2.3. Probabilistic Methods

Related work conducted by Huang and Russell [40, 41] 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 8). 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. Transformation weights should reflect the importance of the rule for the matching of the attribute values. One of the future goals of our system is to learn to adjust these weights to fit the specific domain.

Probabilistic models of the data are also used within the record linkage community [3, 27, 45, 63, 80]. 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 [80, 45] 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.

5. CONCLUSIONS AND FUTURE WORK

This paper has presented the Active Atlas object identification system that addresses the problem of mapping objects between structured web sources, where the same objects can appear in similar yet inconsistent text formats. The experimental results for the system have shown that Active Atlas can achieve high accuracy, while limiting user involvement. For our future work in this area we would like to reduce user interaction even further.

We found that some transformations can be more appropriate for a specific application domain than others; therefore, in order to increase the mapping accuracy, it is important to learn how to weight the transformations appropriately for the application domain. In some domains very specific transformations should be weighted more, e.g. (Abbreviation - "Deli", "Delicatessen"), and in other domains a general transformation function, such as Substring, should be weighted more. In our approach we are planning to combine both the mapping-rule and the transformation weighting learning in order to create a high accuracy object identification system.

The purpose of optimizing the transformation weights is to reduce the number of labeled examples needed by the system. 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.

We plan to combine these two learners (the mapping-rule learner and the transformation weight learner) into one active learning system called the match learner (Figure 28). The match learner will offer the user one example to label at a time, and from that example be able to learn both the mapping rules, as well as, the transformation weights.

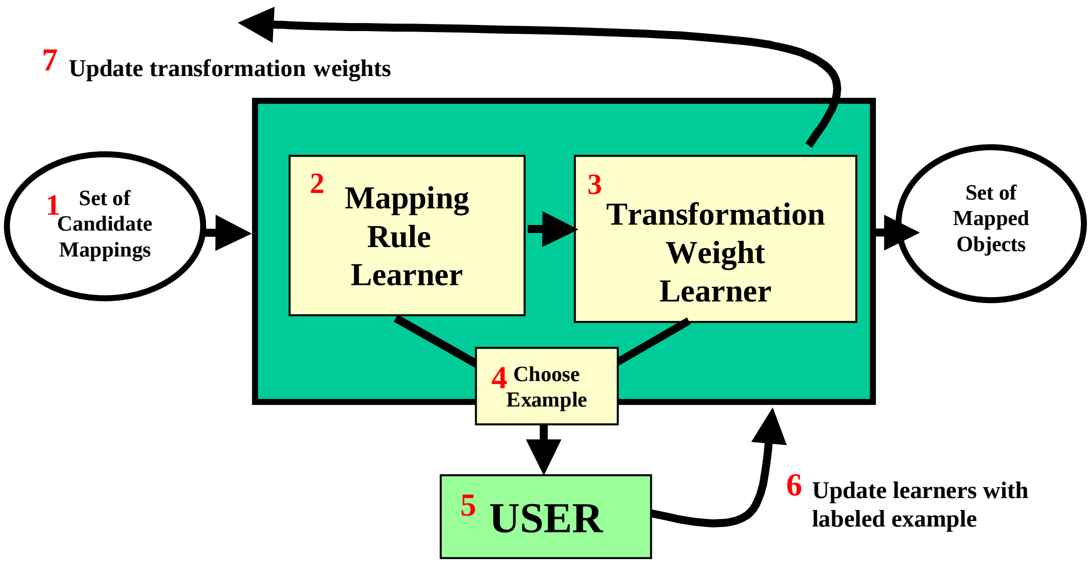

7 Update transformation weights
1 Set of Candidate Mappings
2 Mapping Rule Learner
3 Transformation Weight Learner
Set of Mapped Objects
4 Choose Example
5 USER
6 Update learners with labeled example

Fig. 28: Match Learner

In this hybrid system, there will be two types of learning. One type of learning is the transformation weight learning, which can recognize possible relationships between tokens of object attributes. The second type of learning is mapping-rule learning, which determines the importance of object attributes for generating a mapping. The advantage of this learning system will be that it can intelligently reason and combine information gathered about each of the proposed object mappings in order to decide the total mapping assignment. By simultaneously tailoring both types of learning and efficiently utilizing user input we hope to achieve high accuracy mapping with minimal user input.

Acknowledgements — The research reported here was supported in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-1-0504, in part by the Rome Laboratory of the Air Force Systems Command and the Defense Advanced Research Projects Agency (DARPA) under contract number F30602-98-2-0109, in part by the United States Air Force under contract number F49620-98-1-0046, in part by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152, and in part by a research grant from NCR. 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.

REFERENCES

[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. *Computer* (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] Jose Luis Ambite, Yigal Arens, Naveen Ashish, Craig A. Knoblock, Steven Minton, Jay Modi, Maria Muslea, Andrew Philpot, Wei-Min Shen, Sheila Tejada, and Weixiong Zhang. *The SIMS Manual: Version2.0. Working Draft.* USC/ISI (1997).

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

[6] C. Batini, M. Lenzerini, , and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surverys*, **18**(4):323–364 (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*, pp. 195–206 (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 (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 (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* (2000).

[16] A. Chatterjee and A. Segev. Data manipulation in heterogeneous databases. *SIGMOD Record*, **20**(4) (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* (1997).

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

[20] William W. Cohen. A web-based information system that reasons with structured collections of text. In *Autonomous Agents* (1998).

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

[22] 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* (1998).

[23] 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).

[24] David J. DeWitt, Jeffrey F. Naughton, and Donovan A. Schneider. An evaluation of non-equijoin algorithms. In *VLDB*, pp. 443–452 (1991).

[25] D. Fang, J. Hammer, and D. McLeod. The identification and resolution of semantic heterogeneity in multidatabase systems. *Multidatabase Systems: An Advanced Solution for Global Information Sharing* (1994).

[26] 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*, pp. 82–88 (1996).

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

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

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

[30] 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*, pp. 291–294 (1996).

[31] 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).

[32] 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 (1997).

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

[34] 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* (1996).

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

[36] 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 (1995).

[37] Mauricio Hernandez and Salvatore J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM-SIGMOD Conference* (1995).

[38] Mauricio Hernandez and Salvatore J. Stolfo. A generalization of band joins and the merge/purge problem. *IEEE Transactions on Knowledge and Data Engineering* (1996).

[39] Mauricio Hernandez and Salvatore J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. In *Data Mining and Knowledge Discovery*, pp. 1–31 (1998).

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

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

[42] 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* (1995).

[43] 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* (1995).

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

[45] 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).

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

[47] 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).

[48] 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. *To appear in the International the Journal on Cooperative Information Systems (IJCIS) Special Issue on Intelligent Information Agents: Theory and Applications, Forthcoming* (2001).

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

[50] 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).

[51] 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).

[52] Wen-Syan Li. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the 20th VLDB Conference*, pp. 1–12 (1994).

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

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

[55] 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) (1996).

[56] 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.* (1994).

[57] 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).

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

[59] 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).

[60] 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* (1996).

[61] 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* (1996).

[62] 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* (1997).

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

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

[65] M. Perkowitz and O. Etzioni. Category translation: Learning to understand information on the internet. In *IJCAI'95*, pp. 930–936 (1995).

[66] 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) (1997).

[67] 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* (1998).

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

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

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

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

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

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

[74] 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).

[75] 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* (1995).

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

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

[78] 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.* (1989).

[79] Gio Wiederhold. Intelligent integration of information. *SIGMOD Record*, pp. 434–437 (1993).

[80] William E. Winkler. Matching and record linkage. *Survey Methods for Businesses, Farms, and Institutions* (1994).

[81] Tak W. Yan and Hector Garcia-Molina. Duplicate removal in information dissemination. In *Proceedings of VLDB-95* (1995).

[82] 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).