Learning Domain-Independent String Transformation Weights for High Accuracy Object Identification

Sheila Tejada
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
tejada@isi.edu

Craig A. Knoblock
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 9s0292
knoblock@isi.edu

Steven Minton
Fetch Technologies
4676 Admiralty Way
Marina del Rey, CA 90292
minton@fetch.com

ABSTRACT

The task of object identification occurs 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. Previous methods of object identification have required manual construction of domain-specific string transformations or manual setting of general transformation parameter weights for recognizing format inconsistencies. This manual process can be time consuming and error-prone. We have developed an object identification system called Active Atlas [18], which applies a set of domain-independent string transformations to compare the objects' shared attributes in order to identify matching objects. In this paper, we discuss extensions to the Active Atlas system, which allow it to learn to tailor the weights of a set of general transformations to a specific application domain through limited user input. The experimental results demonstrate that this approach achieves higher accuracy and requires less user involvement than previous methods across various application domains.

1. INTRODUCTION

Information mediators [19], such as SIMS [2] and Ariadne [11], integrate information from multiple information sources on the web. One problem that can arise when integrating information 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. 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.

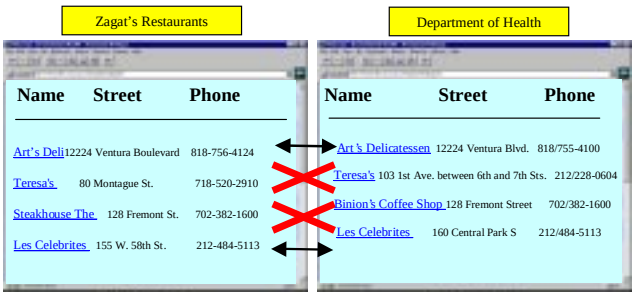

Figure 1: Matching Restaurant Objects

The examples in Figure 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. 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 hotel, so that they both have listed the same address and main phone number of the hotel. Although they match on the **Street** and **Phone** attributes, these restaurants 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

SIGKDD '02 July 23-26, 2002, Edmonton, Alberta, Canada

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.

We have developed an object identification system called Active Atlas [18], which applies a set of general string transformations in order to propose possible mappings between the objects and then employs an active learning technique that learns the necessary mapping rules to classify the mappings with high accuracy. Previously, Active Atlas, as many other object identification systems [15, 20, 14], required that the transformation weights be set manually before the mapping process began.

The focus of this paper is to describe work extending Active Atlas to learn to weight transformations for a specific application domain. 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 a given transformation, like "Equality" or "Acronym," participates in a correct mapping between two objects.

It is expensive, in terms of the user's time, to manually encode this type 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 to classify the data with high accuracy without thoroughly reviewing all data. Yet, in our view the user must play a limited but important role in object mapping 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.

Since the data itself may not be 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. Active Atlas achieves high accuracy object identification by simultaneously learning to tailor both the transformation weights and mapping rules to a specific application domain through limited user input. The main goal of this research is to achieve the highest possible accuracy in object identification with minimal user interaction.

1.1 Active Atlas System Overview

In this paper we provide descriptions of all stages and components of Active Atlas in order to show how transformation weight learning has been integrated into the object identification process. Learning the transformation weights and mapping rules for a specific domain is a circular problem, where the accuracy of the mapping rules depends on the accuracy of the transformation weights and vice-versa. This process has two stages as shown in Figure 2, applying the transformations to calculate initial similarity scores and then learning the mapping rules and transformation weights to properly map the objects between the sources.

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 set of candidate mappings, produced by the candidate generator, serves as the basis for the learning to begin.

In the next stage the mapping learner determines which of the proposed mappings are correct by learning to tailor the appropriate mapping rules and transformation weights to the application domain. The given initial similarity scores are highly inaccurate. To address this problem Active Atlas employs an active supervised learning technique that iteratively refines both the mapping rules and transformation weights. First, mapping rules can be created to classify the mappings, then using this classification information, new transformation weights can be calculated, allowing for new attribute similarity scores and mapping rules that more accurately capture the relationships between the objects.

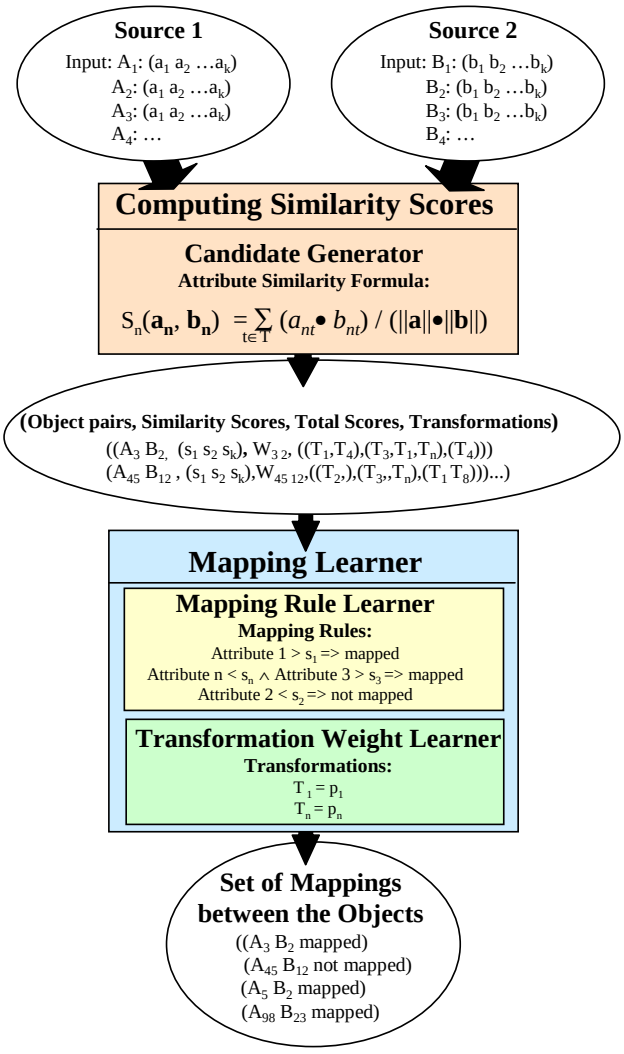

Figure 2: General System Architecture

The remaining sections provide a detailed description of the Active Atlas object identification system. Section 2 explains the process of applying the transformations to compute the initial similarity scores, while section 3 describes

how both learning the transformation weights and mapping rules are integrated together in the mapping learner. Section 4 presents the experimental results of the system. Section 5 discusses related work, and section 6 concludes with a discussion and future work.

2. APPLYING TRANSFORMATIONS

The candidate generator uses a 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 to produce an initial set of quality candidate mappings. The candidate generator keeps a record of the set of transformations that were applied for each mapping, which is essential for learning the transformation weights, and also calculates a set of similarity scores, necessary for learning the mapping rules.

When comparing objects, the alignment of the attributes is determined by the user. The values for each attribute are compared individually (e.g. **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. 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 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 3). The process for computing these scores is described later in this section.

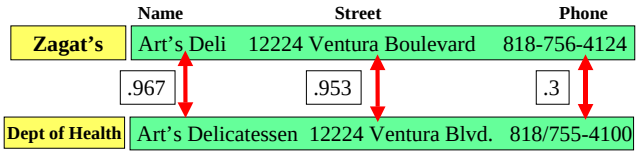

Figure 3: Set of Computed Similarity Scores

Figure 3 displays example attribute similarity scores for the candidate mapping of the objects "Art's Deli" and "Art's Delicatessen." 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 General 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 tokens from two objects. Included in this framework is a set of general domain independent transformations to resolve the different text formats used by the objects. 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.

2.1.1 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 token and replaces with corresponding abbreviation (e.g., 3rd or third).

2.1.2 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

2.2 Generating Candidate Mappings

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 4).

The candidate generator applies the transformations in a three-step process. The first step is to apply all of the unary

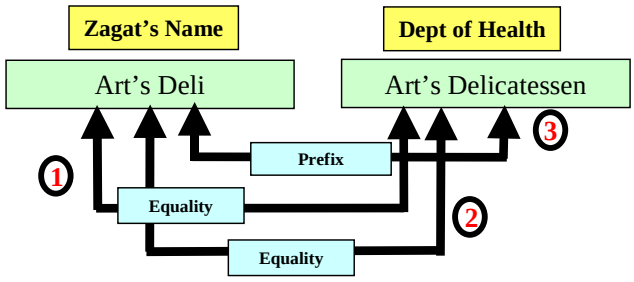

Figure 4: Applying Transformations

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.

Table 1 shows a subset of the candidate mappings for the query object "Art's Deli." This is important information for learning the transformation weights. 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)

Restaurant Name	Hypothesis Rule Set
"Art's Delicatessen"	("Art" Equality "Art") ("s" Equality "s") ("Deli" Prefix "Delicatessen")
"Carnegie Deli"	("Deli" Equality "Deli") (Drop "Carnegie") (Drop "Art") (Drop "s")
"Trattoria Dell'Arte"	("Art" Stemming "Arte") ("Deli" Stemming "Dell") (Drop "Trattoria") (Drop "s")
"Sonora Deli"	("Deli" Equality "Deli") ("s" Initial "Sonora") (Drop "Art")

Table 1: Candidate Mappings with N-ary Transformations

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. Once these sets have been computed for every candidate mapping, then the attribute similarity scores can be calculated.

2.3 Computing Attribute Similarity Scores

The sets of applied transformations are used to calculate the attribute 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 [7] 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 this 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} and w_{ib} correspond to the weights computed by the TFIDF weighting function.

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. For each candidate mapping, the total object similarity score is calculated as a weighted sum of the attribute similarity scores. The mappings are then ranked my their total object similarity score.

3. LEARNING TRANSFORMATION WEIGHTS

The purpose of learning the transformation weights is to reduce the amount of user involvement needed by the system to achieve high accuracy mapping. The candidate generator provides the necessary information to learn the rules and weights for mapping the objects. 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 mapping rules and transformations weights are tailored to a specific application domain

The mapping learner (Figure 5) combines both 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 learner.

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, 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

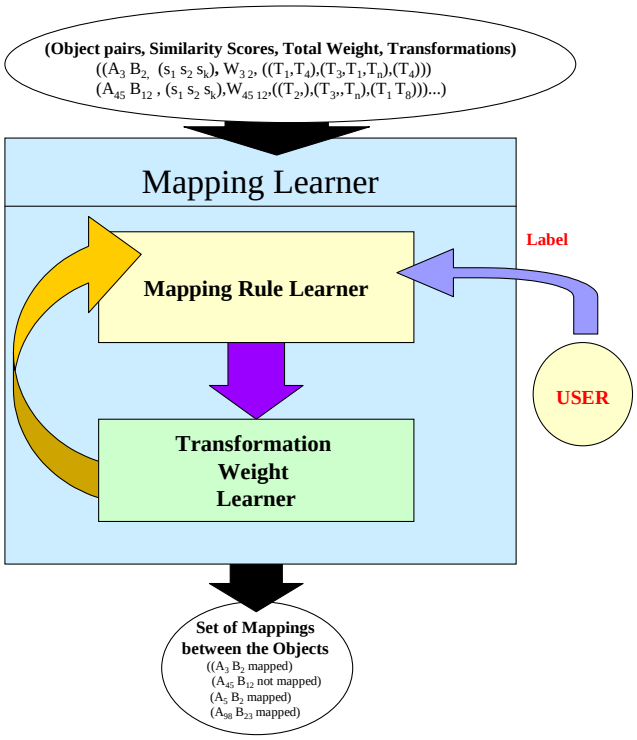

Figure 5: Mapping Learner

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.

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

To efficiently learn the mapping rules for a particular task or domain, we use a supervised learning technique, which uses a combination of several decision tree learners based on an active learning algorithm 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. 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.

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 6). 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 6 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 6: 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.

Once an example is chosen, the user is asked to label the example. 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.

3.2 Transformation Weight Learner

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.

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 new transformation weights takes into consideration classifications of both 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.

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 [16]. 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 if the transformation *tf* is applied between two objects that those objects will be classified as mapped $\hat{m}$ can be estimated using the following formula:

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

$p(m \mid tf_i)$ is calculated for each of the general transformations detailed in section 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.

3.3 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.

$$AttributeSimilarityScore(A, B) = \prod_{i=1}^{t} tf_i$$

Table 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: 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.

4. EXPERIMENTAL RESULTS

In this section we present the experimental results that we have obtained from running Active Atlas with transformation weight learning across three different application domains: Restaurants, Companies and Airports. We have included results from three other object identification system 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, which is similar to previous methods for addressing the merge/purge problem of removing duplicate records in a database [15, 9]. The second system is a baseline experiment that 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 [5]. We will refer to this experiment as the IR system. The third object identification system that will be compared is the previous version of Active Atlas [18],

Zagats	Health Dept	Classification	Labeled by
“Trattoria Dell’Arte”	“Carnegie Deli”	Not Mapped	Learner
“Art’s Deli”	“Art’s Delicatessen”	Mapped	Learner
“Spago (Los Angeles)”	“Spago”	Not Mapped	Learner
“CPK”	“California Pizza Kitchen”	Mapped	User
“Joe’s Restaurant”	”Jan’s Family Restaurant”	Not Mapped	Learner

Table 2: Candidate Mappings with Classifications

which did not perform transformation weight learning and required the transformation weights be set manually prior to the mapping process.

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. 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. 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.

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 [5].

For the IR system experiment the optimal mapping threshold is at rank 111 in the list, and therefore, the top 111 examples in the list are considered mapped together. 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. Also, even though this approach computationally less intensive, it can potentially require more user involvement because a user must view a positive mapping for it to be considered mapped.

The accuracy results from the four 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 7 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 systems require fewer labeled examples than Passive Atlas.

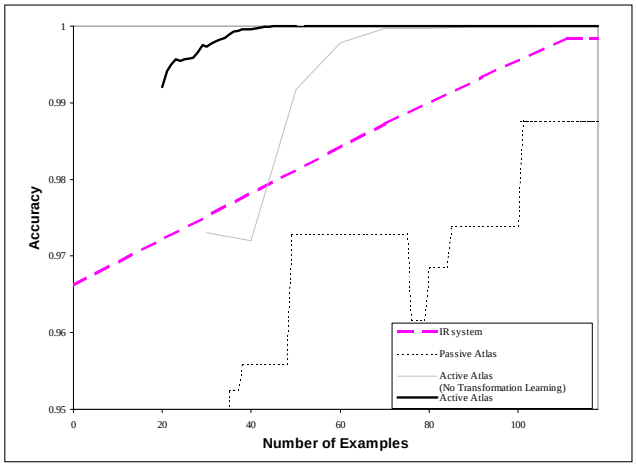

Figure 7: Restaurant Domain Experimental Results

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 both Active Atlas systems can achieve the higher accuracy with fewer labeled examples. The goal of the learning systems is to deduce more information about how the objects match in order to increase the accuracy of the total mapping assignment. Both active learners are 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. The sparse number of positive examples also explains why the IR system accuracy appears high. When the threshold is set to 0 so that no example is considered mapped, it still has over 96% accuracy because the vast majority of examples are not mapped.

These results of the Active Atlas system with transformation weight learning are also a significant improvement (43% fewer examples needed) over the results of Active Atlas with

manually set transformation weights [18]. 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. (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. Therefore, the initial similarity scores calculated by the candidate generator may inaccurately reflect the similarity of the attribute values. The three other approaches must classify examples with inaccurate similarity scores, while the Active Atlas with transformation weight learning can learn to adjust the similarity scores to more accurately capture the true similarity between the attribute values.

4.2 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 8). 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"). Active 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.

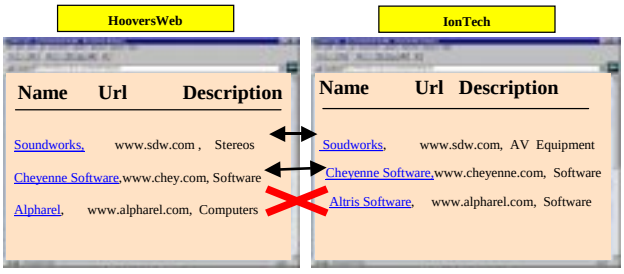

Figure 8: Company Domain Examples

In this domain HooversWeb has 1163 objects and the IonTech site has 957 objects. There are 294 correct mappings between the sites. The optimal threshold for the IR system experiment is at rank 282, where 41 examples are incorrectly classified and 12% of the object mappings are missing. Figure 9 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 9 shows that learning the mapping rules increases the accuracy of the mapping assignment. Active Atlas with transformation weight learning achieves higher accuracy than the IR system experiments at 49 examples (at 7120 examples for Passive Atlas)and 100% accuracy at 95 examples. 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.

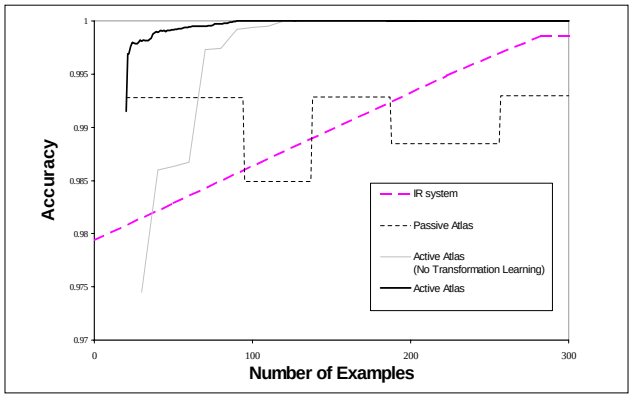

Figure 9: Company Domain Experimental Results

The transformations have more influence in this domain. The transformations are able to resolve the spelling mistake between "Soundworks" and "Soudworks" (Figure 8) using the Soundex transformation, which made the difference in it being mapped or unmapped. Also, these Active Atlas results required 21% fewer examples than Active Atlas with manually set transformation weights.

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 10.

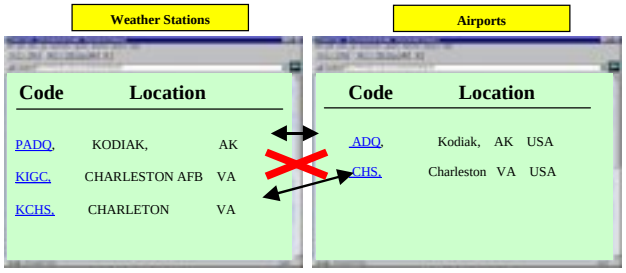

Figure 10: Airport/Weather Domain examples

The optimal threshold for the IR system experiment is set at rank 438, where 220 examples are incorrectly classified and over 25% of the object mappings are missing. 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 11 shows that learning the mapping rules increases the accuracy of the mapping assignment. Active Atlas with transformation weight learning achieves 100% accuracy at 294 examples, and Passive Atlas achieves higher accuracy than the IR system results after only 80 examples. These results are also a significant improvement (40% fewer examples needed) over the results of Active Atlas with manually set transformation weights.

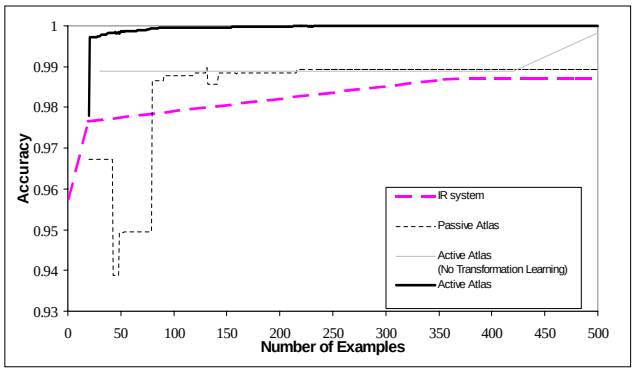

Figure 11: Airport/Weather Domain Experimental Results

5. RELATED WORK

Previous work on object identification has either employed manual methods to customize rules or transformations 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 transformation weights to a specific domain, while limiting user involvement.

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 [4, 3, 12] have been applied by many object identification systems to measure text similarity [8, 9, 10, 21]. The main concern with domain specific transformations is that it is very expensive, in terms of user involvement, to generate a comprehensive set of transformations that are specific not only to the domain, but also to the data that is being integrated.

There are also approaches [15, 17] that use a single very general transformation function, like edit distance, to address the format inconsistencies. These approaches require the user to manually set the transformation parameters for each application domain. Work conducted by Pinheiro and Sun [17] and Ganesh et al. [8] used a supervised learning technique to learn which combinations of attributes are important to generate a mapping. This technique assumes that most objects in the data have at least one duplicate or matching object. It is most similar to the Passive Atlas system, because it requires either that the user choose the examples or they are randomly selected.

The problem of object identification has also appeared in the information retrieval community. 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 [5] 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 is the only transformation used to calculate similarity between strings; therefore, "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 [5, page 9]. Setting a threshold to obtain optimal accuracy is not always a straightforward task for the user. The Citeseer project [13] 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.

Probabilistic models of the data are used within the record linkage community [6, 20]. 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 to normalize attribute values. After the values are normalized then a string edit distance function is applied for which the parameters are manually set. 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 attribute weightings and classify the candidate mappings into one of three classes: mapped, not mapped, or to be reviewed by the user.

Work conducted by McCallum et al. [14] 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.

6. CONCLUSIONS AND FUTURE WORK

We have developed a domain-independent approach for

incorporating the user's knowledge into an object identification system 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. The experimental results demonstrate that Active Atlas achieves higher accuracy and requires less user involvement than previous methods across various application domains.

Currently, we are working on running Active Atlas on larger data sets. There are several issues for future work that we are pursuing, such as providing a method to minimize noise or error in the labels provided by the user. We would also like to applying Active Atlas to other types of related research problems, such as sensor fusion or objection identification for multimedia data.

Acknowledgments

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.

7. REFERENCES

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

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

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

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

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

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

[7] W. Frakes and R. Baeza-Yates. *Information retrieval: Data structures and algorithms*. Prentice Hall, 1992.

[8] 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.

[9] M. Hernandez and S. 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.

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

[11] C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, I. Muslea, A. G. Philpot, and S. 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.

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

[13] S. Lawrence, K. Bollacker, and C. L. Giles. Autonomous citation matching. In *Proceedings of the Third International Conference on Autonomous Agents*, New York, 1999.

[14] A. McCallum, K. Nigam, and L. 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.

[15] A. Monge and C. 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.

[16] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

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

[18] S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Special Issue on Data Extraction, Cleaning, and Reconciliation, Information Systems Journal*, 26(8), 2001.

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

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

[21] T. W. Yan and H. Garcia-Molina. Duplicate removal in information dissemination. In *Proceedings of VLDB*, Zurich, Switzerland, 1995.