Chapter III
A Machine Learning Approach to Data Cleaning in Databases and Data Warehouses

Hamid Haidarian Shahri
University of Maryland, USA

ABSTRACT

Entity resolution (also known as duplicate elimination) is an important part of the data cleaning process, especially in data integration and warehousing, where data are gathered from distributed and inconsistent sources. Learnable string similarity measures are an active area of research in the entity resolution problem. Our proposed framework builds upon our earlier work on entity resolution, in which fuzzy rules and membership functions are defined by the user. Here, we exploit neuro-fuzzy modeling for the first time to produce a unique adaptive framework for entity resolution, which automatically learns and adapts to the specific notion of similarity at a meta-level. This framework encompasses many of the previous work on trainable and domain-specific similarity measures. Employing fuzzy inference, it removes the repetitive task of hard-coding a program based on a schema, which is usually required in previous approaches. In addition, our extensible framework is very flexible for the end user. Hence, it can be utilized in the production of an intelligent tool to increase the quality and accuracy of data.

INTRODUCTION

The problems of data quality and data cleaning are inevitable in data integration from distributed operational databases and online transaction processing (OLTP) systems (Rahm & Do, 2000). This is due to the lack of a unified set of standards spanning over all the distributed sources. One of the
A Machine Learning Approach to Data Cleaning in Databases and Data Warehouses

When data are gathered from distributed sources, differences between tuples are generally caused by four categories of problems in data, namely, the data are incomplete, incorrect, incomprehensible, or inconsistent. Some examples of the discrepancies are spelling errors; abbreviations; missing fields; inconsistent formats; invalid, wrong, or unknown codes; word transposition; and so forth as demonstrated using sample tuples in Table 1.

Very interestingly, the causes of discrepancies are quite similar to what has to be fixed in data cleaning and preprocessing in databases (Rahm & Do, 2000). For example, in the extraction, transformation, and load (ETL) process of a data warehouse, it is essential to detect and fix these problems in dirty data. That is exactly why the elimination of fuzzy duplicates should be performed as one of the last stages of the data cleaning process. In fact, for effective execution of the duplicate elimination phase, it is vital to perform a cleaning stage beforehand. In data integration, many stages of the cleaning can be implemented on the fly (for example, in a data warehouse as the data is being transferred in the ETL process). However, duplicate elimination must be performed after all those stages. That is

<table>
<thead>
<tr>
<th>Discrepancy Problem</th>
<th>Name</th>
<th>Address</th>
<th>Phone Number</th>
<th>ID Number</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spelling Errors</td>
<td>John Doe</td>
<td>Lucent Laboratories</td>
<td>615 5544</td>
<td>553066</td>
<td>Male</td>
</tr>
<tr>
<td>Abbreviations</td>
<td>J. Dow</td>
<td>Lucent Lab.</td>
<td>615 5544</td>
<td>553066</td>
<td>Male</td>
</tr>
<tr>
<td>Missing Fields</td>
<td>John Dow</td>
<td>-</td>
<td>615 5544</td>
<td>-</td>
<td>Male</td>
</tr>
<tr>
<td>Inconsistent formats</td>
<td>John Dow</td>
<td>Lucent Laboratories</td>
<td>(021)6155544</td>
<td>553066</td>
<td>1</td>
</tr>
<tr>
<td>Word Transposition</td>
<td>Dow John</td>
<td>Lucent Laboratories</td>
<td>615 5544</td>
<td>553066</td>
<td>Male</td>
</tr>
</tbody>
</table>

Table 1. Examples of various discrepancies in database tuples
what makes duplicate elimination distinctive from the rest of the data cleaning process (for example, change of formats, units, and so forth).

In order to detect the duplicates, the tuples have to be compared to determine their similarity. Uncertainty and ambiguity are inherent in the process of determining fuzzy duplicates due to the fact that there is a range of problems in the tuples, for example, missing information, different formats, and abbreviations. Our earlier work (Haidarian-Shahri & Barforush, 2004) explored how fuzzy inference can be suitably employed to handle the uncertainty of the problem. Haidarian-Shahri and Barforush described several advantages of the fuzzy expert system over the previously proposed solutions for duplicate elimination. One important advantage is getting rid of the repetitive task of hand-coding rules using a programming language that is very time consuming and difficult to manipulate. This chapter introduces the utilization of neuro-fuzzy modeling on top of the Sugeno method of inference (Takagi & Sugeno, 1985) for the first time to produce an adaptive and flexible fuzzy duplicate elimination framework. Here, we elaborate on how our architecture is capable of learning the specific notion of record similarity in any domain from training examples. This way, the rules become dynamic, unlike hand-coded rules of all the previous methods (Galhardas et al., 2001; Hernandez & Stolfo, 1998; Low et al., 2001; Monge & Elkan, 1997), which in turn assists in achieving better results according to the experiments. Enhancing this novel framework with machine learning and automatic adaptation capabilities paves the way for the development of an intelligent and extendible tool to increase the quality and accuracy of data.

Another chapter of this book by Feil and Abonyi includes an introduction to fuzzy data mining methods. One data cleaning operation may be to fill in missing fields with plausible values to produce a complete data set, and this topic is studied in the chapter by Peláez, Doña, and La Red.

The rest of this chapter is organized as follows. First, we give an account of the related work in the field of duplicate elimination. Then, we describe the design of our architecture. The section after that explains the adaptability and some other characteristics of the framework. Then, we evaluate the performance of the framework and its adaptation capabilities. Finally, we summarize with a conclusion and future directions.

**RELATED WORK**

Generally, data cleaning is a practical and important process in the database industry and different approaches have been suggested for this task. Some of the advantages of our framework over the previous work done on fuzzy (approximate) duplicate elimination are mentioned here. These points will become clearer later as the system is explained in detail in the next sections.

First, the previously suggested fixed and predefined conditions and declarative rules used for comparing the tuples were particularly difficult and time consuming to program (using a programming language), and the coding had to be repeated for different table schemas (Galhardas et al., 2001; Hernandez & Stolfo, 1998; Low et al., 2001; Monge & Elkan, 1997). Our framework uses natural-language fuzzy rules, which are easily defined with the aid of a GUI (graphical user interface). Second, the program (i.e., thresholds, certainty factors [Low et al., 2001] and other parameters) had to be verified again to allow any minor change in the similarity functions. Hence, the hand-coded rules were inflexible and hard to manipulate. Unlike any of the earlier methods, the design of our framework allows the user to make changes flexibly in the rules and similarity functions without any coding.

Third, in previous methods, the rules were static and no learning mechanism could be used in the system. Exploiting neuro-fuzzy modeling equips the framework with learning capabilities.
This adaptation feature is a decisive advantage of the system, and the learning not only minimizes user intervention, it also achieves better results than the user-defined rules, according to the experiments. Therefore, the task of fine-tuning the rules becomes automatic and effortless. Fourth, the design of our system enables the user to easily manipulate different parts of the process and implement many of the previously developed methods using this extendible framework. None of the previous methods take such a comprehensive approach.

Most previous approaches similarly use some form of rules to detect the duplicates. As mentioned above, the design of our framework and deployment of fuzzy logic provides some unique characteristics in our system. The utilization of fuzzy logic not only helps in handling of uncertainty in a natural way, it also makes the framework adaptive using machine learning techniques. Particularly, the learning mechanism in the framework improves performance and was not existent in any of the previous approaches.

SNM (sorted neighborhood method) from Hernandez and Stolfo (1998) is integrated into our framework as well. Hernandez and Stolfo propose a set of rules encoded using a programming language to compare the pairs of tuples. The knowledge-based approach introduced by Low et al. (2001) is similar to our work in the sense of exploiting coded rules to represent knowledge. However, Low et al. do not employ fuzzy inference and use a certainty factor for the coded rules and for the computation of the transitive closures, which is not required here. In our approach, the knowledge base is replaced with fuzzy rules provided by the user. AJAX (Galhardas et al., 2001) presents an execution model, algorithms, and a declarative language similar to SQL (structured query language) commands to express data cleaning specifications and perform the cleaning efficiently. In contrast to our system, these rules are static and hard to manipulate. Nevertheless, the use of a declarative language such as SQL instead of a procedural programming language is very advantageous. Raman and Hellerstein (2001) describe an interactive data cleaning system that allows users to see the changes in the data with the aid of a spreadsheet-like interface. It uses the gradual construction of transformations through examples, using a GUI, but is somewhat rigid; that is, it may be hard to reverse the unwanted changes during the interactive execution. The detection of anomalies through visual inspection by human users is also limiting.

Elmagarmid, Ipeirotis, and Verykios (2007) provide a good and recent survey of various duplicate elimination approaches. Chaudhuri, Ganti, and Motwani (2005) and Ananthakrishna, Chaudhuri, and Ganti (2002) also look at the problem of fuzzy duplicate elimination. Note that the use of the word fuzzy is only a synonym for approximate, and they do not use fuzzy logic in any way. Ananthakrishna et al. use the relations that exist in the star schema structure in a data warehouse to find the duplicates. Sarawagi and Bhamidipaty (2002) use active learning to train a duplicate elimination system; that is, examples are provided by the user in an interactive fashion to help the system learn.

**FLEXIBLE ENTITY RESOLUTION ARCHITECTURE**

Detecting fuzzy duplicates by hand using a human requires assigning an expert who is familiar with the table schema and semantic interpretation of attributes in a tuple; he or she must compare the tuples using expertise and conclude whether two tuples refer to the same entity or not. So, for comparing tuples and determining their similarity, internal knowledge about the nature of the tuples seems essential. Developing a code for this task as proposed by previous methods is very time consuming. Even then, the user (expert) has to deal with parameter tuning of the code by trial and error for the system to work properly.
For finding fuzzy duplicates, Hernandez and Stolfo suggest SNM, in which a key is created for each tuple such that the duplicates will have similar keys. The key is usually created by combining some of the attributes, and the tuples are sorted using that key. The sort operation clusters the duplicates and brings them closer to each other. Finally, a window of size \( w \) slides over the sorted data, and the tuple, entering the window, is compared with all the \( w-1 \) tuples in the window. Hence, performing \( n(w-1) \) comparisons for a total of \( n \) tuples.

A detailed workflow of the duplicate elimination framework is demonstrated in Figure 1. The principal procedure is as follows: to feed a pair of tuples (selected from all possible pairs) into a decision making system and determine if they are fuzzy duplicates or not. First, the data should be cleaned before starting the duplicate elimination phase. That is essential for achieving good results. In a dumb approach, each record is selected and compared with all the rest of the tuples, one by one (i.e., a total of \( n(n-1) \) comparisons for \( n \) records).

To make the process more efficient, the cleaned tuples are clustered by some algorithm in hope of collecting the tuples that are most likely to be duplicates in one group. Then, all possible pairs from each cluster are selected, and the comparisons are only performed for records within each cluster. The user should select the attributes that are important in comparing two records because some attributes do not have much effect in distinguishing a record uniquely. A neuro-fuzzy inference engine, which uses attribute similarities for comparing a pair of records, is employed to detect the duplicates.

This novel framework considerably simplifies duplicate elimination and allows the user to flexibly change different parts of the process. The framework was designed with the aim of producing a user-friendly and application-oriented tool in mind that facilitates flexible user manipulation. In Figure 1, by following the points where the user (expert) can intervene, it is observed that the forthcoming items can be easily selected from a list or supplied by the user (from left to right).

Figure 1. A detailed workflow of the framework
A Machine Learning Approach to Data Cleaning in Databases and Data Warehouses

1. Clustering algorithm
2. Attributes to be used in the comparison of a pair of tuples
3. Corresponding similarity functions for measuring attribute similarity
4. Fuzzy rules to be used in the inference engine
5. Membership functions (MFs)
6. Merging strategy

Most of the above items are explained in this section. Fuzzy rules and membership functions will be explained further in the next sections. Steps 4 and 5 involving the fuzzy rules and membership functions are where the machine learning occurs by using ANFIS (adaptive network-based fuzzy inference system). In this framework, the creation of a key, sorting based on that key, and a sliding window phase of the SNM method is a clustering algorithm. The moving window is a structure that is used for holding the clustered tuples and actually acts like a cluster. The comparisons are performed for the tuples within the window. Any other existing clustering algorithm can be employed, and a new hand-coded one even can be added to the bank of algorithms. For example, another option is to use a priority queue data structure for keeping the records instead of a window, which reduces the number of comparisons (Monge & Elkan, 1997). This is because the new tuple is only compared to the representative of a group of duplicates and not to all the tuples in a group.

The tuple attributes that are to be used in the decision making are not fixed and can be determined dynamically by the user at runtime. The expert (user) should select a set of attributes that best identifies a tuple, uniquely. Then, a specific similarity function for each selected attribute is chosen from a library of hand-coded ones, which is a straightforward step. The similarity function should be chosen according to attribute data type and domain, for example, numerical, string, or domain-dependent functions for addresses, surnames, and so forth. Each function is used for measuring the similarity of two corresponding attributes in a pair of tuples. In this way, any original or appropriate similarity function can be easily integrated into the fuzzy duplicate elimination framework. The fuzzy inference engine combines the attribute similarities and decides whether the tuples are duplicates or not using the fuzzy rules and membership functions as explained in Haidarian-Shahri and Barforush (2004) and Haidarian-Shahri and Shahri (2006). The details are related to how we use the Mamdani method of inference.

At the end, the framework has to eliminate the detected duplicates by merging them. Different merging strategies can be utilized as suggested in the literature (Hernandez & Stolfo, 1998), that is, deciding on which tuple to use as the prime representative of the duplicates. Some alternatives are using the tuple that has the least number of empty attributes, using the newest tuple, prompting the user to make a decision, and so on. All the merged tuples and their prime representatives are recorded in a log. The input-output of the fuzzy inference engine (FIE) for the detected duplicates is also saved. This information helps the user to review the changes in the duplicate elimination process and verify them. The rule viewer enables the expert to examine the input-output of the FIE and fine-tune the rules and membership functions in the framework by hand, if required.

FRAMEWORK CHARACTERISTICS

Due to the concise form of fuzzy if-then rules, they are often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision. The variables are partitioned in terms of natural-language linguistic terms. This linguistic partitioning, an inherent feature of what Lotfi Zadeh (2002) calls computing with words, greatly simplifies model building. Linguistic terms represent fuzzy subsets over the
corresponding variable’s domain. These terms are what we actually use in our everyday linguistic reasoning as we speak. Consequently, the rules can be easily defined by the expert.

It has been shown that the decision-making process is intrinsically difficult, taking into account the ambiguity and uncertainty involved in the inference. It is also time consuming and quite impossible to assign a human to this task, especially when dealing with large amounts of data. The system has a robust design for fuzzy duplicate elimination, and has several interesting features, as explained here.

**Adaptation and Learning Capabilities**

The fuzzy reasoning approach provides a fast and intuitive way of defining the rules by the expert in natural language with the aid of a simple GUI. This eliminates the repetitive process of hard-coding and reduces the development time. An example of a rule in this framework is as follows: IF (LastNameSimilarity is high) ∧ (FirstNameSimilarity is high) ∧ (CodeSimilarity is high) ∧ (Address-Similarity is low) THEN (Probability = 0.9). In this rule, LastNameSimilarity is a linguistic variable and high is a linguistic term that is characterized by an MF. The definition of linguistic variable can be found in Zadeh (1975a, 1975b, 1975c) and in another chapter of this book, written by Xexeo. Generally, the antecedent part of each rule can include a subset (or all) of the attributes that the user has selected previously. The consequence or output of the rule represents the probability of two tuples being duplicates.

In the rules for the Mamdani method of inference (Mamdani, 1976), the output variable is fuzzy. The Mamdani method is utilized when the rules and hand-drawn MFs are defined by the user without any learning and adaptation. This method is more intuitive and suitable for human input. Humans find it easier to state the rules that have fuzzy output variables, such as Probability = high. On the other hand, in the rules for the Sugeno method of inference (Takagi & Sugeno, 1985), the output variable is defined by a linear equation or a constant, for example Probability = 0.85. This is computationally more efficient and works better with adaptive techniques. Hence, learning can be applied on top of a Sugeno fuzzy inference system (FIS). In grid partitioning or subtractive clustering (as explained in this section), the user only determines the number of membership functions for each input variable to form the initial structure of the FIS. This way, there is no need to define any rules or MFs by hand. The adaptation mechanism will handle the rest, as we will explain.

Fuzzy rules specify the criteria for the detection of duplicates and the rules effectively capture the expert’s knowledge that is required in the decision-making process. In our system, the only tricky part for the expert is to determine the fuzzy rules and membership functions for the inference engine. By taking advantage of neuro-fuzzy techniques (Jang & Sun, 1995) on top of the Sugeno method of inference, the framework can be trained using the available numerical data, which mitigates the need for human intervention. The numerical data used for training are vectors. Each vector consists of the attribute similarities of a pair of tuples (inputs of the fuzzy inference engine) and a tag of zero or one (output of the FIE) that determines whether the pair is a duplicate or not. Note that the results of employing the Mamdani method of inference, which merely employs the rules provided by the user in natural language without any learning, is quite acceptable as presented in Haidarian-Shahri and Barforush (2004). Adding adaptation and learning capabilities to the framework enhances the results, as shown in the experiments. Later, we will explain more about the training process and the number of training examples that are required. The details of the adaptation process are provided in Jang and Sun.
A Machine Learning Approach to Data Cleaning in Databases and Data Warehouses

The process of constructing a fuzzy inference system is called fuzzy modeling, which has the following features.

• Human expertise about the decision-making process is integrated into the structure determination of the system. This usage of domain knowledge is not provided by most other modeling methods. Structure determination includes determining the relevant inputs, the number of MFs for each input, the number of rules, and the type of fuzzy model (e.g., Mamdani, Sugeno).

• When numerical input-output data for the system to be modeled are available, other conventional system identification methods can be employed. The term neuro-fuzzy modeling refers to applying learning techniques developed in the neural networks literature to parameter identification of FISs. Parameter identification deals with recognizing the shape of MFs and the output of rules, to generate best performance.

By employing ANFIS, the membership functions are molded into shape and the consequence of the rules are tuned to model the training data set more closely (Jang, 1993; Jang & Sun, 1995). The ANFIS architecture consists of a five-layered adaptive network, which is functionally equivalent to a first-order Sugeno fuzzy model. This network (i.e., fuzzy model) can be trained when numerical data are available. The adaptation of the fuzzy inference system using machine learning facilitates better performance. When numerical input-output data are not available, the system merely employs the rules provided by the user in natural language as explained in Haidarian-Shahri and Barforush (2004).

In essence, the spirit of a fuzzy inference system is “divide and conquer”; that is, the antecedents of fuzzy rules partition the input space into a number of local fuzzy regions, while the consequents describe the behavior within a given region. In our experiments, grid partitioning (Bezdek, 1981) and subtractive clustering (Chiu, 1994) are used to divide (partition) the problem space and determine the initial structure of the fuzzy system. Then ANFIS is applied for learning and fine-tuning of the parameters. Grid partitioning uses similar and symmetric MFs for all the input variables to generate equal partitions without clustering. The subtractive clustering method partitions the data into groups called clusters and generates an FIS with the minimum number of rules required to distinguish the fuzzy qualities associated with each of the clusters.

Two methods are employed for updating the membership function parameters in ANFIS learning: (a) back-propagation (BP) for all parameters (a steepest descent method), and (b) a hybrid method consisting of back-propagation for the parameters associated with the input membership functions, and least-squares estimation for the parameters associated with the output membership functions. As a result, the training error decreases in each fuzzy region, at least locally, throughout the learning process. Therefore, the more the initial membership functions resemble the optimal ones, the easier it will be for the parameter training to converge.

The most critical advantage of the framework is its machine learning capabilities. In previous methods used for duplicate elimination, the expert had to define the rules using a programming language (Hernandez & Stolfo, 1998; Low et al., 2001). The task of determining the thresholds for the rules and other parameters, like the certainty factor, was purely done by trial and error (Low et al.). In this system, not only is the hard-coding, but the system also adapts to the specific meaning of similarity based on the problem domain using the provided training examples. Even in cases when numerical data for training is unavailable, the framework can be utilized using the membership functions and simple commonsense rules provided by the expert to achieve acceptable performance. It is very valuable to consider that, although there
might be other learning mechanisms that are feasible to be utilized for this task, none are likely to be so accommodative and user-friendly to allow the framework (tool) to operate with and without training data. Haidarian-Shahri and Barforush (2004) report on the use of the system and handling of uncertainty without any training.

Note that, here, the learning is done at a meta-level to capture the specific notion of record similarity, which is the quantity that needs to be measured for the detection of fuzzy duplicate records. This is more than developing trainable similarity functions for specific types of fields or domain-independent similarity functions. In fact, this framework allows the user to employ any previously developed and complex learnable string similarity measure (Bilenko & Mooney, 2003; Monge & Elkan, 1997) in the duplicate elimination process, as shown in Step 3 of Figure 1.

**Other Features**

Other features of the framework are briefly described here. More details can be found in Haidarian-Shahri and Barforush (2004) and Haidarian-Shahri and Shahri (2006). When the expert is entering the rules, he or she is in fact just adding the natural and instinctive form of reasoning as if performing the task by hand. Here, the need for the time-consuming task of hard-coding a program and its parameter tuning is eliminated. Additionally, by using fuzzy logic, uncertainty is handled inherently in the fuzzy inference process, and there is no need for a certainty factor for the rules.

The user can change different parts of the framework, as previously illustrated in Figure 1. Consequently, duplicate elimination is performed very flexibly. Since, the expert determines the clustering algorithm, tuple attributes, and corresponding similarity functions for measuring their similarity, many of the previously developed methods for duplicate elimination can be integrated into the framework. Hence, the framework is quite extendible and serves as a platform for implementing various approaches.

Obviously, domain knowledge helps the duplicate elimination process. After all, what are considered duplicates or data anomalies in one case might not be in another. Such domain-dependent knowledge is derived naturally from the business domain. The business analyst with subject-matter expertise is able to fully understand the business logic governing the situation and can provide the appropriate knowledge to make a decision. Here, domain knowledge is represented in the form of fuzzy rules, which resemble humans’ way of reasoning under vagueness and uncertainty. These fuzzy if-then rules are simple, structured, and manipulative.

The framework also provides a rule viewer and a logging mechanism that enables the expert to see the exact effect of the fired rules for each input vector, as illustrated in Haidarian-Shahri and Barforush (2004) and Haidarian-Shahri and Shahri (2006). This, in turn, allows the manipulation and fine-tuning of problematic rules by hand, if required. The rule viewer also provides the reasoning and explanation behind the changes in the tuples and helps the expert to gain a better understanding of the process.

**PERFORMANCE AND ADAPTATION EVALUATION**

For implementing the fuzzy duplicate elimination framework, the Borland C++ Builder Enterprise Suite and Microsoft SQL Server 2000 are used. The data reside in relational database tables and are fetched through ActiveX Data Object (ADO) components. The Data Transformation Service (DTS) of MS SQL Server is employed to load the data into the OLE DB Provider. The hardware setup in these experiments is a Pentium 4 (1.5 GHz) with 256 MB of RAM and the Windows XP operating system.
A Machine Learning Approach to Data Cleaning in Databases and Data Warehouses

The data set used in our experiments is made up of segmented census records originally gathered by Winkler (1999) and also employed in some of the previous string matching projects (Cohen, Ravikumar, & Fienberg, 2003). The data are the result of the integration of two different sources, and each source has duplicates as well as other inconsistencies. The table consists of 580 records, of which 332 are unique and 248 are duplicates. The records are very similar to the ones shown in Table 1. For the purpose of these experiments and investigating the effectiveness of the approach, a simple similarity function is used in our implementation, which only matches the characters in the two fields and correspondingly returns a value between zero and one. That is, the more characters two strings have in common, the more their similarity would be. This is basically using the Jaccard string similarity measure. For two strings s and t, the similarity measure would return the ratio of intersection of s and t to the union of s and t. However, by adding smarter and more sophisticated attribute similarity functions that are domain dependant (handling abbreviations, address checking, etc.), the final results can only improve. The fuzzy inference process is not explained here and the reader can refer to Mamdani (1976), Takagi and Sugeno (1985), and Haidarian-Shahri and Shahri (2006) for more details.

Four attributes, namely, last name, first name, code, and address, are selected by the expert and employed in the inference process. The basic SNM is used for the clustering of records. Two linguistic terms (high and low) are used for the bell-shaped hand-drawn membership functions of the input variables, as shown in Figure 2 (left), which allows for the definition of a total of $2^4$ rules. The output variable consists of three linguistic terms (low, medium, high), as demonstrated in Figure 2 (right). Humans find it easier to state the rules that have fuzzy output variables, as in the Mamdani method. The expert adds 11 simple rules, similar to the following, in natural language, with the aid of a GUI.

- IF (LastNameSimilarity is low) ∧ (FirstNameSimilarity is high) ∧ (CodeSimilarity is high) ∧ (AddressSimilarity is high) THEN (Probability is medium).
- IF (LastNameSimilarity is low) ∧ (FirstNameSimilarity is low) THEN (Probability is low).

To evaluate the performance of the approach and adaptation effectiveness, recall and precision are measured. Recall is the ratio of the number of retrieved duplicates to the total number of duplicates. False-positive error ($FP_e$) is the ratio of the number of wrongly identified duplicates to the total number of identified duplicates. Precision is equal to $1-FP_e$. Obviously, the performance of the system is better if the precision is higher at a given recall rate. The precision-recall curves

*Figure 2. Linguistic terms (low, high) and their corresponding membership functions for the four input variables (on the left), and the linguistic terms (low, medium, high) for the output variable (on the right)*
for hand-drawn membership functions and the FISs resulting from applying ANFIS are shown in Figure 3. For all the cases in this figure, two linguistic terms (membership functions) are used for each input variable.

Several hand-drawn bell-shaped MFs are tested using different crossing points for the low and high terms. The best results are achieved with the crossing point at 0.6 as shown in Figure 2, and the curve labeled “Best Hand-Drawn” and plotted in Figure 3 is for that shape. When using adaptation, the initial structure of the FIS is formed using grid partitioning, and the user does not specify any rules or MFs. Different combinations of hybrid and back-propagation learning on bell and Guassian shapes are experimented with and it is observed that the trained FISs perform better than the FIS using hand-drawn MFs and user-defined rules. Hybrid learning on Gaussian MFs shows the best performance and achieves a 10 to 20% better precision at a given recall rate.

In Figure 3, note that by using a very simple hand-drawn shape and primitive rules defined by the expert, the system is able to detect 70% of the duplicates with 90% precision without any programming. The resultant data are more accurate and quite acceptable (Haidarian-Shahri & Barforush, 2004). By employing learning, the framework even achieves better results, successfully detecting 85% of the duplicates with 90% precision. The data set used for the training consists of the comparisons performed for a window size of 10. A total of 5,220 comparisons are recorded and the duplicates are marked. Each vector in the training data set consists of the four attribute similarities and a tag of zero (not duplicate) or one (duplicate). This data set is broken into three equal parts for training, testing, and validation. In the ANFIS training process, the FIS is trained using the training data set, the error rate for the validation data set is monitored, and parameters, which perform best on the validation data set (not the training data set), are chosen for the inference system. Then the FIS is tested on the testing data set. This way, model overfitting on the training data set, which degrades the overall performance,

Figure 3. Comparison of user-generated and grid-partitioned FISs using different combinations of learning and MF shapes
is avoided. Cross-validation ensures that the system learns to perform well in the general case, that is, for the unseen data.

As our initial conjecture (Haidarian-Shahri & Barforush, 2004), the experiments showed that using more than two linguistic terms (high and low) for input variables does not improve the results because when the similarity of attributes (as measured by the user-selected function) is not high, the actual similarity value and the difference between the attributes are of no significance. Hence, there is no need for more than two terms. Having two linguistic terms also limits the total number of possible rules.

Figure 4 show the effect of ANFIS learning on the consequence part of the rules (decision surface) at epochs 0, 10, and 26. The training was performed for 30 epochs, and these epoch

Figure 4. The effect of ANFIS learning on the consequence (z-value) part of the rules; the learning algorithm gradually produces a decision surface that matches the training data.

Figure 5. Comparison of grid partitioning and subtractive clustering for the initial structure of an FIS and using different learning methods.
numbers are chosen to demonstrate the gradual change of the decision surface during the training procedure. It illustrates the underlying effect of training on the dynamic fuzzy rules. Note that the ANFIS is capable of adapting to produce a highly nonlinear mapping between input and output in the \(n\)-dimensional problem space. Here, two dimensions are shown: the hybrid learning algorithm, and grid partitioning to perform the initial division of the problem input space. The consequence (\(z\)-value) of a rule determines the probability of a pair of tuples being duplicates. As demonstrated in Figure 4, rule consequences are all set to zero at the start, and the learning algorithm gradually produces a decision surface that matches the training data as much as possible, reducing the error rate. Figure 4 is showing the change of the first-name and last-name input variables, marked as input3 and input4, respectively. The tuples have four attributes, namely, code, address, first name, and last name.

Figure 5 demonstrates the precision-recall curve for the best trained FIS resulting from grid partitioning and the FISs generated using subtractive clustering with hybrid and back-propagation learning. Here, subtractive clustering uses five MFs per input variable. The performance is similar for the three cases in the figure. Therefore, subtractive clustering is also quite effective for partitioning.

By employing a set of simple rules, easily worded in natural language by the user who is familiar with the records, acceptable results are achieved. In this approach, very little time is spent on phrasing the rules, and the burden of writing hard-code with complex conditions is mitigated. This is not a surprise because intuitiveness and suitability for human comprehension is the inherent feature of fuzzy logic. To top that off, when training data are available, our design exploits neuro-fuzzy modeling to allow users to de-duplicate their integrated data adaptively and effortlessly. This even alleviates the need for specifying obvious rules and regular membership functions.

**CONCLUSION**

In this chapter, we introduce a novel and adaptive framework for de-duplication. Essentially, it would not be possible to produce such a flexible inference mechanism without the exploitation of fuzzy logic, which has the added benefit of removing time-consuming and repetitive programming. Utilizing this reasoning approach paves the way for an easy-to-use, accommodative, and intelligent duplicate elimination framework that can operate with or without training data. Therefore, with this framework, the development time for setting up a de-duplication system is reduced considerably. The results show that the system is capable of eliminating 85% of the duplicates at a precision level of 90%.

The advantages of utilizing fuzzy logic in the framework for fuzzy duplicate elimination include the ability to specify the rules in natural language easily and intuitively (domain knowledge acquisition), the ability to remove the hard-coding process, framework extendibility, fast development time, flexibility of rule manipulation, inherent handling of uncertainty of the problem without using different parameters, and most importantly, adaptability. If training data are not available, duplicate elimination is done using the natural-language rules and membership functions provided by the user. Furthermore, if training data are available, the use of ANFIS and machine learning capabilities virtually automates the production of the fuzzy rule base and specification of the membership functions.

All together, these features make the framework very suitable and promising to be utilized in the development of an application-oriented commercial tool for fuzzy duplicate elimination, which is our main future goal. Perhaps another interesting future line of work is to implement this approach using standard fuzzy data types and the clustering technique defined in (Galindo, Urrutia, & Piattini, 2006), which defines some fuzzy data types and many fuzzy operations on
these values using FSQL (fuzzy SQL) with 18 fuzzy comparators, like FEQ (fuzzy equal), NFEQ (necessarily FEQ), FGT (fuzzy greater than), NF GT (necessarily FGT), MGT (much greater than), NMGT (necessarily MGT), inclusion, and fuzzy inclusion.

ACKNOWLEDGMENT

The authors would like thank helpful comments and suggestions by Dr. Galindo and anonymous reviewers, which increased the quality of this chapter.

REFERENCES


A Machine Learning Approach to Data Cleaning in Databases and Data Warehouses


**KEY TERMS**

**Data Cleaning:** Data cleaning is the process of improving the quality of the data by modifying their form or content, for example, removing or correcting erroneous data values, filling in missing values, and so forth.

**Data Warehouse:** A data warehouse is a database designed for the business intelligence requirements and managerial decision making of an organization. The data warehouse integrates data from the various operational systems and is typically loaded from these systems at regular intervals. It contains historical information that enables the analysis of business performance over time. The data are subject oriented, integrated, time variant, and nonvolatile.

**Machine Learning:** Machine learning is an area of artificial intelligence concerned with the development of techniques that allow computers to learn. Learning is the ability of the machine to improve its performance based on previous results.

**Mamdani Method of Inference:** Mamdani’s fuzzy inference method is the most commonly seen fuzzy methodology. It was proposed in 1975.
by Ebrahim Mamdani as an attempt to control a steam engine and boiler combination. *Mamdani-type inference* expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. It is possible, and in many cases much more efficient, to use a single spike as the output membership function rather than a distributed fuzzy set. This type of output is sometimes known as a *singleton* output membership function, and it can be thought of as a “predefuzzified” fuzzy set. It enhances the efficiency of the defuzzification process because it greatly simplifies the computation required by the more general Mamdani method, which finds the centroid of a two-dimensional function. Rather than integrating across the two-dimensional function to find the centroid, you use the weighted average of a few data points. Sugeno-type systems support this type of model.

**OLAP (Online Analytical Processing):** OLAP involves systems for the retrieval and analysis of data to reveal business trends and statistics not directly visible in the data directly retrieved from a database. It provides multidimensional, summarized views of business data and is used for reporting, analysis, modeling and planning for optimizing the business.

**OLTP (Online Transaction Processing):** OLTP involves operational systems for collecting and managing the base data in an organization specified by transactions, such as sales order processing, inventory, accounts payable, and so forth. It usually offers little or no analytical capabilities.

**Sugeno Method of Inference:** Introduced in 1985, it is similar to the Mamdani method in many respects. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator, are exactly the same. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant.