

A Flexible Fuzzy Expert System for Fuzzy Duplicate Elimination in Data Cleaning

Hamid Haidarian Shahri and Ahmad Abdollahzadeh Barforush

Faculty of Computer Engineering and Information Technology
Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
hhaidarian@aut.ac.ir, ahmad@ce.aut.ac.ir

Abstract. Data cleaning deals with the detection and removal of errors and inconsistencies in data, gathered from distributed sources. This process is essential for drawing correct conclusions from data in decision support systems. Eliminating fuzzy duplicate records is a fundamental part of the data cleaning process. The vagueness and uncertainty involved in detecting fuzzy duplicates make it a niche, for applying fuzzy reasoning. Although uncertainty algebras like fuzzy logic are known, their applicability to the problem of duplicate elimination has remained unexplored and unclear, until today. In this paper, a novel and flexible fuzzy expert system for detection and elimination of fuzzy duplicates in the process of data cleaning is devised, which circumvents the repetitive and inconvenient task of hard-coding. Some of the crucial advantages of this approach are its flexibility, ease of use, extendibility, fast development time and efficient run time, when used in various information systems.

1 Introduction

The problems of data quality and *data cleaning* are inevitable in data integration from distributed operational databases and OLTP systems [9]. That is due to the lack of a unified set of standards spanning over all the distributed sources. One of the most challenging and resource intensive phases of data cleaning is the removal of fuzzy duplicate records. The term “fuzzy duplicates” is used for tuples, which are somehow different, but describe the same real-world entity, i.e. different syntaxes but the same semantic. *Eliminating* fuzzy duplicates is applicable in any database, but critical in data integration and analytical processing domains, where accurate reports/statistics are required. Some of the application domains of fuzzy duplicate elimination include data warehouses especially in the dimension tables, OLAP and data mining applications, decision support systems, on-demand (lazy) web-based information integration systems, web search engines, meta-crawlers and numerous others. Consequently, a flexible approach to detect the duplicates can be utilized in many *database applications*.

In this paper, section 2 describes the design of a fuzzy expert system. Section 3 explains the characteristics of the system. Section 4 contains the experimental results and their evaluation. Section 5 gives an account of the related work in the field of duplicate elimination and section 6 summarizes with a conclusion.

2 Design of the Fuzzy Duplicate Elimination System

Detecting fuzzy duplicates using a human and by hand, requires assigning an expert, who is familiar with the table schema and semantic interpretation of attributes in a tuple, for comparing the tuples using his expertise and concluding, 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. Generally, an “equational theory” is utilized to compare two tuples and infer the probability of them being the same [5]. Considering the uncertainties involved, sometimes the decision making is indeed difficult, for a human as well. It is very practical and beneficial to automate this long and cumbersome task, by replacing the human with an expert system.

For finding fuzzy duplicates, Hernandez suggests the *sorted neighborhood method* (SNM) [5], in which a key is created for each tuple, such that the duplicates will have similar keys. 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. In previous methods in the literature, the comparison is done using a set of static and predefined conditions or declarative rules that are hard-coded, which is different from this approach, and much more time consuming.

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. A detailed workflow of the duplicate elimination system is demonstrated in Figure 1. Firstly, 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 gathering 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 selects the *attributes* that are important in comparing two records, because some attributes do not have much effect in distinguishing a record uniquely. A fuzzy expert system, which uses attribute similarities for comparing a pair of records, is employed to detect the duplicates.

This novel duplicate elimination system allows the user to *flexibly* change different parts of the process. In Figure 1, by following the points, where the user 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): 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, 6) merging strategy. Most of these items are explained in this section. Fuzzy rules and membership functions will be explained further in the next sections.

In this system, the creation of a key, sorting based on that key, and sliding window phase of the SNM method, is considered as a clustering algorithm. The moving window is a structure, which is used for holding the clustered tuples and actually acts

like a cluster. The comparisons are performed for tuples within the window. Any other existing *clustering algorithm* can be employed, and even a new hand coded one 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 [8].

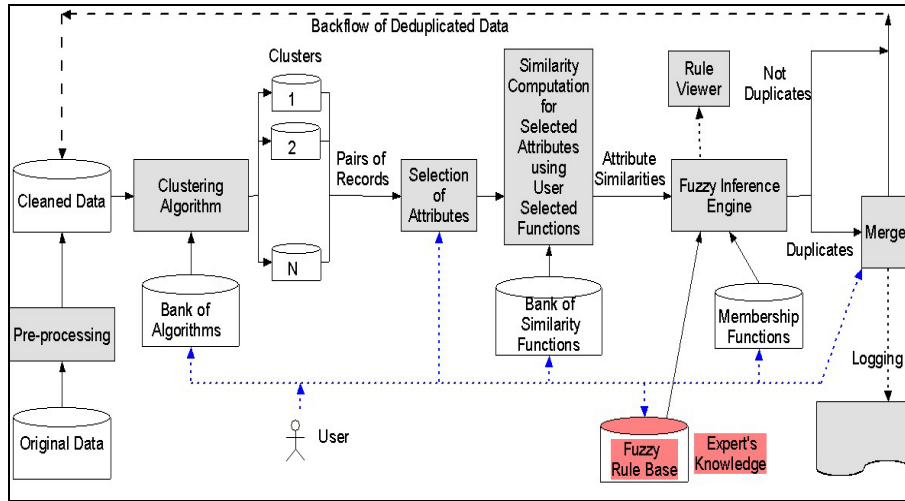


Fig. 1. A detailed workflow of the fuzzy duplicate elimination system

The *tuple attributes* that are to be used in the decision making are not fixed and can be determined by the user, at run-time. The expert (user) should select a set of attributes, which best identifies a tuple, uniquely. Hence, the number of attributes in the tuple is not very critical. Then, a specific *similarity function* for each selected attribute is chosen, from a library of hand coded ones, which is a straight-forward step. The similarity function should be chosen according to attribute data type and domain, e.g. numerical, string, or domain dependent functions for address, surname, etc. 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 system.

At the end, the system has to eliminate the detected duplicates, by merging them. Different *merging strategies* can be utilized, as suggested in the literature [5], i.e. deciding on which tuple to use, as the prime representative of the duplicates. Some alternatives are using the tuple, which has the least number of empty attributes, the newest tuple, or prompting the user to make a decision, etc. 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 system by hand. The rule viewer has been implemented and is described in [4].

3 Characteristics of the System

This design for the fuzzy duplicate elimination system, as described in the previous section, has several interesting features that are explained here in more detail. The decision making process is intrinsically difficult considering the ambiguity and uncertainty involved in the inference. It is also time consuming and quite impossible to assign humans to this task, especially when dealing with large amounts of data. The solution here is a flexible fuzzy expert system.

3.1 Acquisition of Expert's Knowledge

In this system, fuzzy rules specify the criteria for the detection of duplicates. In fact, the rules provide a simple way of utilizing any complex logic for computing text similarity and string manipulations, in terms of similarity functions selected by the user. The rules effectively *capture expert's knowledge*, which is required in the decision making process. The essence of reasoning based on fuzzy rules [7] is briefly explained here. An example of a fuzzy rule that describes a simple fact is: IF *pressure* is *high* THEN *volume* is *small*, where pressure and volume are linguistic variables, and high and small are linguistic terms that are characterized by *membership functions*. 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 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.

In contrast to the fuzzy reasoning approach of this system, the previously suggested conditions and declarative rules for comparing tuples are particularly difficult and time consuming to code and the coding has to be repeated for each different table schema. 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 system is as follows: IF (LastNameSimilarity is *high*) \wedge (FirstNameSimilarity is *high*) \wedge (CodeSimilarity is *high*) \wedge (AddressSimilarity is *low*) THEN (Probability is *medium*). The antecedent part of each rule can include a *subset* (or *all*) of the attributes, which the user has selected in the previous stage. The consequence or output of the rule represents the probability of two tuples being duplicates.

In defining the membership function for each linguistic (input) variable, e.g. LastNameSimilarity, two linguistic terms (low, high) are used. When the similarity of attributes (as measured by the user-selected function) is not high, the actual similarity value and how much different the attributes are, are of no importance. Hence, there is no need for more than two terms. Having two linguistic terms also limits the number of possible rules. The shape of the membership function for each linguistic term is determined by the expert.

3.2 Handling Uncertainty

When the expert is entering the rules, he is in fact just adding his natural and instinctive form of reasoning, as if he was to perform this task by hand. However, in previous methods, there are much more problems, when thresholds and certainty factors for rules [6] and other parameters, have to be hard-coded. The previously proposed certainty factor represents the belief of the expert, who is coding the rules, in the effectiveness of a rule. By using fuzzy logic, *uncertainty* is handled differently in the *fuzzy inference* process and there is no need for a certainty factor. Fuzzy inference inherently handles the uncertainty involved in the decision making, because it measures the *degree* to which an instance fits a certain rule. Additionally, here, the problem of parameter tuning of the hard-coded program, which is quite time consuming and a matter of trial and error, is alleviated.

Generally, the fuzzy-system model determines the value of the consequent variable, for a given manifestation of the antecedent variables. In essence, a fuzzy-system model is a knowledge-based representation of the functional relationship, between the antecedent variables and the consequent variable. Such a model's inference mechanism is straight-forward, as formalized below. In Mamdani method of inference [7], the consequent variable is a fuzzy subset, for example, a rule ends with something like Probability = high, etc. If the final consequent of the fuzzy inference system is above a certain threshold, the tuples are classified as duplicates. The actual and appropriate value for the threshold should be determined, according to the consequence of the rules. [4] explains the inference with more details and an example.

Consider that there are n rules and the i^{th} rule is: IF V_1 is A_{i1} & V_2 is A_{i2} & ... & V_p is A_{ip} , THEN U is B_i , where V_i 's are the antecedent variables and U is the consequent variable. A_{ij} and B_i are fuzzy subsets over the corresponding variable's domain. With this representation for a fuzzy rule, the output for a given input, $V_j = x_j^*$, is determined in the following process:

1. Calculate each rule's firing level as, $\lambda_i = \text{Min}_{j=1, \dots, p} \{A_{ij}(x_j^*)\}$. Here, $A_{ij}(x_j^*)$ is the membership grade of x_j^* in the fuzzy subset A_{ij} .
2. Calculate each rule's effective output by weighting its consequence by its firing level. This process produces the fuzzy subset F_i , where $F_i(y) = \lambda_i B_i(y)$.
3. Aggregate the effective outputs to obtain a fuzzy subset F corresponding to the overall system output, where $F(y) = \text{Max}_{j=1, \dots, n} [F_j(y)]$.
4. Calculate the crisp specific model output y^* from F , using a defuzzification step. This produces a unique value for the consequent variable.

3.3 Other Features

The user can change different parts of the process, as previously illustrated in Figure 1. Consequently, duplicate elimination is performed very freely and *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 system. For example, the

user can employ complex learnable string similarity measures [1] or domain independent ones [8], etc. Hence, the system is quite *extendible* and serves as a *framework* 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 human's way of reasoning under vagueness and uncertainty. These fuzzy if-then rules are simple, structured and *manipulative*.

An expert fuzzy inference engine applies the rules to the input and quickly performs the reasoning using basic arithmetic operations that require little computational power. The *computational efficiency* provided by this approach is arguably higher than the code developed in [5] and the expert system suggested in [6] using the Rete algorithm. The reason is that here the rules are all fired in parallel and the processing is a basic min or max operation, unlike previous methods. There is no need to run a code step by step. This is a crucial advantage, when comparing millions of tuples in a large database. The system also provides a rule viewer that enables the expert to see the exact effect of the fired rules for each input vector. This in turn, allows the manipulation and fine-tuning of problematic rules by hand. 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.

4 Experimental Results and Evaluation

For implementing the fuzzy duplicate elimination system, the Borland C++ Builder Enterprise Suite and Microsoft SQL Server 2000 are used. The data resides in relational database tables and is 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 in these experiments is a Pentium 4 (1.5 GHz) with the Windows XP operating system. The dataset used in our experiments is made up of segmented census records originally gathered by Winkler [11] and also employed in some of the previous string matching projects [2]. The data is the result of 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 structure of records and some examples from the actual dataset are shown in Table 2 below.

As illustrated in the examples above, some attributes in a record can be null. The similarity function returns a code for those attributes and consequently those attributes are not used in the comparison of the two records. For the purpose of these experiments and investigating the effectiveness of the approach, a very 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. However, by adding more sophisticated and smarter attribute similarity functions that are

domain dependant (handling abbreviations, address checking, etc.), the final results can only improve.

Table 1. Structure of records and examples from the actual dataset

| Record Source | Last Name | First Name | Middle Initial | Code | Address |
|---------------|-----------|------------|----------------|------|---------|
| A | RUSSALL | MYRIAM | A | 624 | OCONEE |
| A | RUSSELL | MIRIAM | M | 624 | OCONEE |
| A | JIMENCZ | IRVIN | A | - | BANK |
| B | JIMENCZ | AUNDERE | I | 214 | BANK |

Here, the process is explained for the first two records in the table above. Four of the attributes as selected by the user (last name, first name, code and address) are employed in the inference process. The basic SNM is exploited for the clustering of records. Two linguistic terms of high and low are used for the bell-shaped membership functions of the input variables in this example, as shown in Figure 2 (left), which allows for the definition of a total of 4^2 rules. The output variable consists of three linguistic terms (low, medium, high), as demonstrated in Figure 2 (right). The user adds eleven rules, similar to the following, in natural language, with the aid of a GUI:

- IF (LastNameSimilarity is *high*) \wedge (FirstNameSimilarity is *high*) \wedge (CodeSimilarity is *high*) \wedge (AddressSimilarity is *high*) THEN (Probability is *high*).
- IF (LastNameSimilarity is *low*) \wedge (FirstNameSimilarity is *low*) THEN (Probability is *low*).

The input vector of the fuzzy inference engine, as calculated by the simple attribute similarity function, is: (0.857, 0.958, 1, 1) and the unique output produced by the rules using the Mamdani method is 0.939, which is above the 0.8 threshold set by the user for this test and hence, the tuples are detected as duplicates, which is actually correct. Due to space constraints, the inference process is not explained any further and the reader can refer to [7] for more details. The user can efficiently *visualize* the effect of rules in the inference process with the aid of the logging mechanism and the rule viewer.

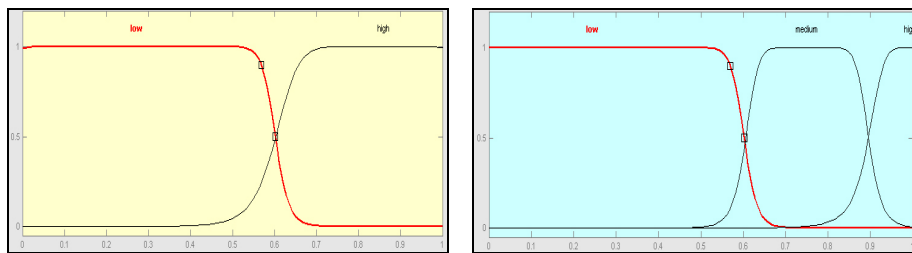


Fig. 2. Linguistic terms (low, medium, high) and their corresponding membership functions for the four input variables (on the left), and the output variable (on the right)

Figure 3 is showing the effect of window size, and different keys that are employed in sorting the data in the SNM method, in single-pass and multi-pass, with the eleven

rules of the previous example. Using an unsuitable key, which is not able to group the duplicates together, has a deterring effect on the result, i.e. more false duplicates are detected than true ones, using the address key with a window size of 3. Hence, key creation is important in the SNM. Generally, as in the figure, the increase in window size enhances the results, but not much is achieved by using a very large window, which increases the execution time and number of comparisons. In multi-pass, when several runs of the process is repeated using a different key each time, only a small window suffices and with a window size of 3, the rules effectively identify 79% of the duplicates with a precision of 85%. The resultant data is of good quality, with considerably less duplicates, i.e. 195 of the 248 duplicates are eliminated.

Figure 4 demonstrates the precision-recall curve for the above rules with a window size of 6 using multi-pass, which describes the trade off between the critical parameters of precision and recall. By following the curve, it can be inferred that the system performed quite well on the dataset, effectively finding 175 of the 248 duplicates at one point, which is a recall rate of 70 percent, while sustaining a 90 percent precision. By employing other sets of simple rules, *worded* easily in natural language by the user, who is familiar with the records, similar results have been achieved. Overall, it is observed that the system is capable of handling the ambiguity and uncertainty involved, in the cumbersome task of fuzzy (approximate) duplicate elimination. In this approach, very little time is spent on phrasing the rules, which mitigates the burden of writing hard-code with complex conditions that has to be repeated for different database schemas. This is not a surprise, because it is the inherent feature of fuzzy logic. However, our design for the duplicate elimination system, exploits that feature to allow users to de-duplicate their integrated data, effortlessly. The system provides flexibility for the user to change or manipulate various parts of the process and implement many of the existing methods for this task.

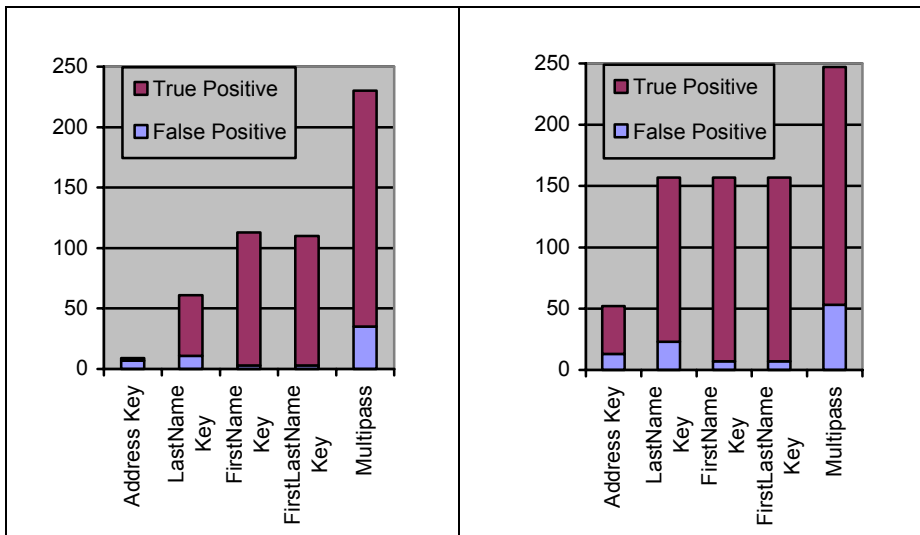


Fig. 3. The effect different keys in single-pass and multi-pass with window sizes of 3 on the left and 10 on the right hand side. 248 duplicates exist in the dataset of 580 records

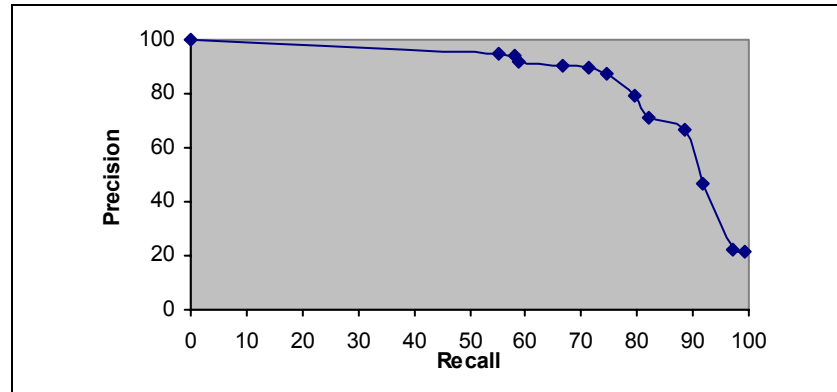


Fig. 4. The precision-recall curve for the eleven rules with a window size of 6 using multi-pass

5 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 differences of our system, to the previous work done on approximate duplicate elimination, have been mentioned throughout the paper, especially in section three. This design can be used for integrating many of the previously suggested methods in the literature, like various clustering algorithms, similarity functions and mergers. Additionally, utilizing a fuzzy expert system introduces some new beneficial features. The seminal SNM method from [5] and the idea of an equational theory have been utilized in our de-dup system. The knowledge-based approach introduced by Lee [6] is similar to our work, in the sense of exploiting hand-coded rules to represent knowledge. [6] suggests using a certainty factor, for the rules and for computing the transitive closures, which is not required here.

[10] describes an interactive data cleaning system, that allows the user to see the changes in the data with the aid of a spreadsheet-like interface. It uses gradual construction of transformations through examples, to eliminate the need for programming. AJAX [3] presents an execution model, algorithms and a declarative language, similar to SQL commands, to express data cleaning specifications and perform the cleaning efficiently.

6 Conclusions and Future Work

This paper introduces the application of *fuzzy logic* for de-duplication for the first time. 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 programming from the production of an “equational theory.” Utilizing this reasoning approach paves the way, for an easy to use and accommodative duplicate

elimination system, which actually implements many of the previous methods. Hence, the development time for setting up a de-dup system is reduced considerably. Another advantage of the Mamdani method of inference is computational efficiency. The inference produces a unique value for the consequent, using a simple computation, and computational efficiency is important for providing the rapid response time needed in performing comparisons, when evaluating many tuples. The experiments reveal that, in multi-pass SNM, the rules effectively identify 79% of the duplicates with a precision of 85%, using a very small window size of 3, which significantly reduces the number of comparisons and run-time.

The advantages of utilizing fuzzy logic for fuzzy duplicate elimination include: the ability of specifying the rules in natural language easily and intuitively, removing the hard-coding process, fast development time, flexibility of rule manipulation, and computational efficiency of the inference process. These features make the approach very suitable and promising to be utilized, in the development of an *application-oriented* commercial tool for duplicate elimination, which is our main future plan. Discovering more about the rules that are required, based on the attributes selected by the user, and perhaps automating the rule production, is another avenue for further research. The possibility of adding learning mechanisms to the system looks very promising as well, and is going to be investigated.

References

1. Bilenko, M., Mooney, R.J.: Adaptive Duplicate Detection Using Learnable String Similarity Measures. Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03), Washington, DC, August (2003)
2. Cohen, W., Ravikumar, P., Fienberg, S.: A Comparison of String Distance Metrics for Name-Matching Tasks. In IIWeb Workshop 2003 (2003)
3. Galhardas, H., Florescu, D., et al.: Declarative Data Cleaning: Language, Model, and Algorithms. In Proc. of the 27th VLDB Conference (2001)
4. Haidarian Shahri, H., Barforush, A.A.: Data Mining for Removing Fuzzy Duplicates Using Fuzzy Inference. 23rd International Conference of the North American Fuzzy Information Processing Society (NAFIPS 2004), Banff, Alberta, Canada, June 27-30 (2004)
5. Hernandez, M.A., Stolfo, S.J.: Real-world Data is Dirty: Data Cleansing and the Merge/Purge Problem. *Data Mining and Knowledge Discovery* 2 (1) (1998) 9–37
6. Low, W.L., Lee, M.L., Ling, T.W.: A Knowledge-based Approach for Duplicate Elimination in Data Cleaning. *Information Systems* 26 (2001) 585-606
7. Mamdani, E.H.: Advances in Linguistic Synthesis of Fuzzy Controllers. *Int. J. Man Machine Studies*, Vol. 8 (1976)
8. Monge, A.E., Elkan, P.C.: An Efficient Domain-independent Algorithm for Detecting Approximately Duplicate Database Records. Proceedings of the SIGMOD 1997 Workshop on Data Mining and Knowledge Discovery, May (1997)
9. Rahm, E., Do, H.H.: Data Cleaning: Problems and Current Approaches. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Special Issue on Data Cleaning*, 23(4) December (2000)
10. Raman, V., Hellerstein, J.M.: Potter's Wheel: An Interactive Data Cleaning System. In Proc. of the 27th VLDB Conference (2001)
11. Winkler, W.E.: The State of Record Linkage and Current Research Problems. *Statistics of Income Division, Internal Revenue Service Publication R99/04* (1999)