LEARNING-BASED FUSION FOR DATA DEDUPLICATION:
A ROBUST AND AUTOMATED SOLUTION

by

Jared Dinerstein

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

Approved:

Dr. Stephen W. Clyde
Major Professor

Dr. Vicki Allan
Committee Member

Dr. Stephen Allan
Committee Member

Dr. Byron R. Burnham
Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2010
Copyright © Jared Dinerstein 2010

All Rights Reserved
ABSTRACT

Learning-Based Fusion for Data Deduplication:
A Robust and Automated Solution

by

Jared Dinerstein, Master of Science
Utah State University, 2010

This thesis presents two deduplication techniques that overcome the following critical and long-standing weaknesses of rule-based deduplication: (1) traditional rule-based deduplication requires significant manual tuning of the individual rules, including the selection of appropriate thresholds; (2) the accuracy of rule-based deduplication degrades when there are missing data values, significantly reducing the efficacy of the expert-defined deduplication rules.

The first technique is a novel rule-level match-score fusion algorithm that employs kernel-machine-based learning to discover the decision threshold for the overall system automatically. The second is a novel clue-level match-score fusion algorithm that addresses both Problem 1 and 2. This unique solution provides robustness against missing/incomplete record data via the selection of a best-fit support vector machine. Empirical evidence shows that the combination of these two novel solutions eliminates two critical long-standing problems in deduplication, providing accurate and robust
results in a critical area of rule-based deduplication.
ACKNOWLEDGMENTS

I would like to thank my advisor, Stephen Clyde, for all of his patience, help, and support.

I would also like to thank my wife, Meg, for her unfailing encouragement. Thank you to my family.

Jared Dinerstein
## CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT ................................................................. iii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS .................................................. v</td>
</tr>
<tr>
<td>LIST OF TABLES ......................................................... x</td>
</tr>
<tr>
<td>LIST OF FIGURES ......................................................... xi</td>
</tr>
</tbody>
</table>

### CHAPTER

1. INTRODUCTION ................................................................. 1

2. THE DEDUPLICATION PROBLEM ........................................ 5
   - Previous Deduplication Work ........................................ 5
     - Matching ................................................................. 8
     - Linking/Merging ......................................................... 9
   - Blocking ................................................................. 9
   - Classification .......................................................... 10
   - Previous Learning-Based Deduplication Techniques ........ 14
     - Freely Extensible Biomedical Record Linkage .......... 15
     - Global Justice XML Data Model .............................. 16
     - Chapter Summary .................................................. 16

3. RULE-LEVEL FUSION FOR DATA DEDUPLICATION .............. 18
   - Motivation ............................................................... 18
   - Overview of Technique ............................................. 19
     - Rule Description ................................................... 20
       - Primary Rules ..................................................... 20
       - Secondary Rules .................................................. 24
     - Traditional (Non-Fused) Deduplication .................... 24
     - Rule-Level Fused Deduplication .............................. 26
     - Detailed Description .............................................. 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probe Record</td>
<td>27</td>
</tr>
<tr>
<td>Filters</td>
<td>27</td>
</tr>
<tr>
<td>Primary Rules</td>
<td>28</td>
</tr>
<tr>
<td>Clue Functions</td>
<td>29</td>
</tr>
<tr>
<td>Clues</td>
<td>30</td>
</tr>
<tr>
<td>Secondary Rules</td>
<td>30</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>33</td>
</tr>
<tr>
<td>Rule-Level Matching Machine</td>
<td>33</td>
</tr>
<tr>
<td>Rationale for Training the SVM</td>
<td>34</td>
</tr>
<tr>
<td>Steps for Training the SVM</td>
<td>34</td>
</tr>
<tr>
<td>Generating the Training Examples</td>
<td>34</td>
</tr>
<tr>
<td>Selecting the Training Examples</td>
<td>36</td>
</tr>
<tr>
<td>Selecting the SVM Parameters</td>
<td>37</td>
</tr>
<tr>
<td>Rule-Level SVM Fusion</td>
<td>39</td>
</tr>
<tr>
<td>Experiment</td>
<td>39</td>
</tr>
<tr>
<td>Data Sets</td>
<td>39</td>
</tr>
<tr>
<td>Method</td>
<td>42</td>
</tr>
<tr>
<td>Primary Rules</td>
<td>42</td>
</tr>
<tr>
<td>Secondary Rules</td>
<td>45</td>
</tr>
<tr>
<td>Rule Match-Score Level Fusion</td>
<td>47</td>
</tr>
<tr>
<td>Testing Data</td>
<td>48</td>
</tr>
<tr>
<td>Testing the Rule-Level Fused Deduplication</td>
<td>49</td>
</tr>
<tr>
<td>Empirical Results</td>
<td>51</td>
</tr>
<tr>
<td>Discussion of Strengths and Weaknesses</td>
<td>60</td>
</tr>
<tr>
<td>4. ROBUST DEDUPLICATION WITH MISSING RECORD DATA</td>
<td>63</td>
</tr>
<tr>
<td>Motivation</td>
<td>63</td>
</tr>
<tr>
<td>Overview of Technique</td>
<td>64</td>
</tr>
<tr>
<td>Detailed Description</td>
<td>70</td>
</tr>
<tr>
<td>Clue Set</td>
<td>70</td>
</tr>
<tr>
<td>Best-Fit Support Vector Machines</td>
<td>70</td>
</tr>
<tr>
<td>Clue-Level Fusion</td>
<td>72</td>
</tr>
<tr>
<td>Clue-Level Matching Machine</td>
<td>74</td>
</tr>
<tr>
<td>Rationale for Training the Bank of Best-Fit SVMs</td>
<td>75</td>
</tr>
<tr>
<td>Steps for Training the SVM</td>
<td>75</td>
</tr>
<tr>
<td>Generating the Training Examples</td>
<td>75</td>
</tr>
</tbody>
</table>
Rule-Level Fusion Versus Clue-Level Best-Fit Multi-SVM Fusion........118
Overview of Other Matchers and Discussion........................................120

6. CONCLUSIONS................................................................................................. 122

Implications............................................................................................. 123
Summary ................................................................................................. 124
Future Work............................................................................................. 126

REFERENCES ............................................................................................................... 128
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>FEBRL-generated Data Sets</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>FEBRL-generated Data Sets</td>
<td>82</td>
</tr>
<tr>
<td>5.1</td>
<td>The Data Sets Used for the Additional Experiments</td>
<td>103</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>This figure shows some generic and basic data points with no hyperplane separating the groups of data, where one category is made of Xs and the second category of Os.</td>
</tr>
<tr>
<td>2.2</td>
<td>This figure shows a representative maximum hyperplane for the categories presented in Figure 2.1. The dotted line separates the data points into two separate classes; the solid lines represent the margin between the two classes.</td>
</tr>
<tr>
<td>2.3</td>
<td>This figure shows a non-maximized hyperplane, when compared to Figure 2.2. This hyperplane is over-fitted to the problem space.</td>
</tr>
<tr>
<td>2.4</td>
<td>This figure shows two classes of data that are not linearly separable.</td>
</tr>
<tr>
<td>2.5</td>
<td>GJXDM XML fragment. GJXDM is a standard U.S. government schema containing person-identifying elements, such as surname, given name, and the Soundex code of both names.</td>
</tr>
<tr>
<td>3.1</td>
<td>Pseudo code representation of rule-level fused deduplication technique.</td>
</tr>
<tr>
<td>3.2</td>
<td>Visual representation of the rule-level fusion matching machine.</td>
</tr>
<tr>
<td>3.3</td>
<td>The primary rules generate a candidate set, using a gallery of records as the source for the candidate set.</td>
</tr>
<tr>
<td>3.4</td>
<td>A generic rule, containing some number (n) of clues and clue functions.</td>
</tr>
<tr>
<td>3.5</td>
<td>A high-level view of the (non-fused) rule-based deduplication system.</td>
</tr>
<tr>
<td>3.6</td>
<td>A generic example showing the relationship between rules, match-scores and an appropriately trained SVM classifier, when performing rule-level fusion.</td>
</tr>
<tr>
<td>3.7</td>
<td>A basic filter.</td>
</tr>
<tr>
<td>3.8</td>
<td>A basic Primary Rule. This basic rule contains a single filter that creates the primary candidate set from the original gallery of records.</td>
</tr>
<tr>
<td>3.9</td>
<td>A clue function that uses Soundex to determine the similarity between two text fields.</td>
</tr>
</tbody>
</table>
3.10 This basic clue example shows the generic functionality of a clue used by a rule……………………………………………………………………………….. 32
3.11 A Secondary Rule example……………………………………………… 33
3.12 A training example for multi-rule fusion………………………………... 36
3.13 Two-stage, rule-based deduplication……………………………………… 43
3.14 Primary Rule 1. Primary rule 1 is the strictest of the four primary rules, and relies on strongly identifying fields such as passportID…………….. 43
3.15 Primary Rule 2. This more lenient rule is applied only if the first primary rule does not return any candidates…………………………………….. 44
3.16 Primary Rule 3. This rule is even more lenient than the previous primary rules: for most fields, it only requires a match in the first character…………………………………………………………………. 44
3.17 Primary Rule 4. Primary rule 4 is the most lenient of the four primary rules………………………………………………………………………. 45
3.18 Secondary Rule 1. The secondary rules calculate the record match scores. This rule calculates the Levenshtein edit distance of identifying fields, or clues, such as passportID……………………………………… 46
3.19 Secondary Rule 2. This rule utilizes a higher number of clues containing less strongly identifying fields than secondary rule 1……… 46
3.20 Secondary Rule 3. This rule contains ambiguous fields, such as proper names, so this rule has slightly looser clue-level thresholds……………. 47
3.21 This figure displays the concatenation of single rule match-scores. The SVM predicts the classification (match/no match) of the probe-candidate pair……………………………………………………………………. 50
3.22 This figure shows the general outline of conditions to outcome for displaying the experimental results produced by an RL-MM……………… 52
3.23 The deduplication results for the first data set presented in Table 3.1 (with 89.77% data set similarity). This data set contains 3646 duplicates and 1495354 non-duplicates (for all 1000 probe and 1500 gallery records)……………………………………………………………………………… 54
3.24 The deduplication results for the second data set presented in Table 3.1 (with 72.00% data set similarity). This data set contains 3644 duplicates and 1495356 non-duplicates (for all 1000 probe and 1500 gallery records)------------------------------- 54

3.25 The deduplication results for the third data set presented in Table 3.1 (with 52.81% data set similarity). This data set contains an average total of 3608.6 duplicates and 1495391.3 non-duplicates (for all 1000 probe and 1500 gallery records)------------------------------- 55

3.26 Overall view of the sensitivity, data similarity and matching percentage achieved for each data set-------------------------------------------- 59

4.1 A generic rule containing clues and clue functions----------------------------- 66

4.2 A generic representation of the clue-level best-fit SVM deduplication…… 67

4.3 We train a set of best-fit SVMs for the clues contained in Secondary Rule, SR (shown in Figure 4.2)----------------------------------------------- 67

4.4 A high-level, overall view of the best-fit multi-SVM clue-level fusion technique described in this chapter-------------------------------------- 69

4.5 This figure shows distinct clue combinations created from one secondary rule's clue set----------------------------------------------- 70

4.6 Logic diagram representing clue-level fusion--------------------------------- 74

4.7 This figure details how traditional non-fused deduplication uses clue-level match-scores (and the clue-level thresholds) to specify a classification for records contained in the primary candidate set……………… 76

4.8 This figure shows how clue-level fusion leverages the clues within a secondary rule. The SVM determines the classification of the current clue match-scores-------------------------------------------- 77

4.9 A general example of the format of the training data for clue-level fusion----------------------------------------------- 78

4.10 Primary Rule 1. The primary rules generate the primary candidate set… 84

4.11 Primary Rule 2---------------------------------------------------------- 85

4.12 Primary Rule 3---------------------------------------------------------- 85

4.13 Primary Rule 4---------------------------------------------------------- 86

4.15 Secondary Rule 2.

4.16 Secondary Rule 3. This rule contains ambiguous fields, such as proper names.

4.17 This figure represents the format of the data inputted into the best-fit SVM. The SVM predicts the classification (match/no match) of the probe-candidate pair.

4.18 Experimental results with no missing data. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).

4.19 Experimental results with 15% missing data. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).

4.20 Experimental results with 30% missing data. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).

4.21 Experimental results with 45% missing data. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).

4.22 The different techniques applied to each data set shown in Table 4.1, with the x-axis emphasizing the percentage of missing data in each experimental data set.

5.1 Primary Rule 1 in Set A.

5.2 Primary Rule 2 in Set A.

5.3 Secondary Rule 1 in Set A.

5.4 Secondary Rule 2 in Set A.

5.5 Primary Rule 1 in Set B.

5.6 Primary Rule 2 in Set B.

5.7 Primary Rule 3 in Set B.

5.8 Secondary Rule 1 in Set B.
5.9  Secondary Rule 2 in Set B
5.10  Secondary Rule 3 in Set B
5.11  The results obtained by Rule Set A with 30% missing data and a similarity of 54.38%. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).
5.12  The results obtained by Rule Set A with 14% missing data and a similarity of 72%. This data set contains a total of 3644 duplicates and 1495356 non-duplicates (for all 1000 probe and 1500 gallery records).
5.13  This figure shows Rule Set A applied to both data sets: one data set containing 30% missing data (with 52.38% similarity), and a second data set with 14.03% missing data (with 72% similarity).
5.14  Deduplication results obtained by Rule Set B with 30% missing data and 54.38% data set similarity. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).
5.15  Deduplication results obtained by Rule Set B with 14% missing data and 72% data set similarity. This data set contains a total of 3644 duplicates and 1495356 non-duplicates (for all 1000 probe and 1500 gallery records).
5.16  Deduplication results of Rule Set B on both data sets: one data set containing 30% missing data (with 52.38% similarity), and a second data set with 14.03% missing data (with 72% similarity).
Duplicate data, in the domain of computer data management, is any data that is stored at more than one location in a computer system, including duplicate data stored on the same computer, in the same network, or even in external memory. This is true not only of data that is stored two or more times in its entirety, but also partially repeated data. A classic example is data records (stored in a database) that contain identical or partially overlapping information. The duplicate data problem follows from the extreme storage capabilities of modern computers, wherein a commodity hard disk can easily store the equivalent of thousands of encyclopedia volumes. Among such large quantities of data, it is commonplace for unrecognized duplicates to be present. This duplicate data problem is a current, critical, and yet unsolved, problem in database systems. Duplicate data has a negative impact on data storage and retrieval, system execution performance, data quality, and the fundamental usefulness of the data. For example, in a relational database, duplicate data causes unnecessary bloating of the data set, consuming secondary storage resources and requiring search queries to take extra, unwarranted CPU cycles. Note that duplication can improve access time, for example, if a record is stored under two different keys or in multiple tables. However, despite this speed increase, duplicate data tends to cause problems when updating database entries. Moreover, duplicate data records are often incomplete, with only the union of all overlapping records providing a complete understanding of the data element. Thus, duplicate data reduces the efficacy of a database system as a whole.
There is need for effective and automatic deduplication of data records, which is a challenging and multi-faceted problem that requires: (1) automatic detection of two overlapping records, and (2) merging of partial information from each record to form a new, more complete record. Both steps are difficult due to the varied nature of data to which deduplication could be applied. While existing approaches, as detailed in Chapter 2, present various alternatives for dealing with data deduplication, none of them have been able to propose a solution that successfully resolves this problem in its entirety. While others performed important work on data deduplication, as detailed in the Previous Work section, there are no sufficient solutions.

This thesis presents a robust, flexible, and automated solution for duplicate record detection based on rule-based data deduplication that utilizes learning-based information fusion, specifically in kernel-machine-based learning [1, 2, 3]. Our novel method applies individual deduplication rules to data record fields, and then combines the resultant match scores via learning-based information fusion into one overall match score. This aggregate match score effectively determines whether two data records represent the same entity. Our learning-based fusion design (1) alleviates the need for manual tuning of the deduplication rules, (2) provides robust detection of overlapping records beyond what individual rules can provide, and (3) is highly reliable when faced with partial or missing record data.

Unlike existing deduplication approaches, our proposed technique has a number of key strengths, including: (1) automatic and rapid tuning of the deduplication rules, (2) robust detection of overlapping records that is an order of magnitude more accurate than traditional methods, and (3) reliability when faced with partial or missing record data.
This novel solution represents a significant step forward in the state-of-the-art for automated duplicate record detection. We demonstrate this fused, learning-based deduplication on a significant real-world schema.

Rule-based deduplication suffers from two primary weaknesses: (1) effective identification of duplicate data requires significant manual tuning of the deduplication rules [3], and (2) poor duplicate detection occurs when there are missing data values [3]. This thesis consists of two complementary techniques that address these weaknesses.

The first technique, presented in Chapter 3, is a rule-based deduplication technique that automatically tunes the expert-provided deduplication rules by using machine-learning techniques, removing the need for extensive manual tuning. Rule-based deduplication can produce accurate results, but this accuracy is dependent on the efficacy of the included rules and typically requires significant manual tuning. This process is not only time-consuming but is limited to the domain knowledge of a human expert, leaving the system suboptimal, and resulting in ambiguous deduplication results that require manual clerical review [3]. In these cases, a human analyst must perform a visual inspection to determine whether two data records represent an actual match [3]. Even though manual tuning of the deduplication rules can reduce the need for this clerical review, by improving the efficacy and accuracy of the expert-defined rules, the benefits reaped by repetitive improvements to the rules become increasingly minute for the time trade-off. The approach presented in Chapter 3 eliminates the need for this extensive, manual tuning.

The second technique, presented in Chapter 4, is a distinctly separate deduplication technique based on multi-support vector machine (SVM) fusion that
addresses the real-world problem of accurately detecting and identifying duplicate records in data sets wherein missing or partial data is prevalent [4]. This multi-SVM fusion technique employs a bank of support vector machines, each of which can be applied to a given subset of fields to produce accurate deduplication of data records that contain empty fields, while continuing to address the issues identified in Chapter 3, including eliminating the need for extensive, manual tuning of the expert-provided rules. Thus, multi-SVM fusion provides accurate deduplication results while robustly addressing missing data.
CHAPTER 2

THE DEDUPLICATION PROBLEM

Data deduplication, i.e., record linkage or record matching, is not a new problem [3, 5, 6, 7]. In fact, from the late 1950s through the early 1960s, Newcombe et al. performed extensive foundational research on duplicate record detection [6, 8]. At its core, the most fundamental aspect of data deduplication is *duplicate record detection*, which is the process either of identifying database records that exist in the same database, or in disparate databases, that actually refer to the same real-world entity, even if the database records are not explicitly identical. There is a critical need for better and more efficient record deduplication as the accidental duplication of data is common in many problem spaces and domains [3].

Previous Deduplication Work

During the last fifty years, researchers have identified the following four approaches to deduplicating record sets: probabilistic matching models, e.g., Bayesian inference, supervised and semi-supervised learning, active learning systems (ALS), and distance-based techniques [3]. Two groups or fields of research make up these techniques.

The first group of deduplication techniques is made up of those techniques that require sets of training data upon which these techniques can be trained (or *learn*) to identify matching records. These are *principled* approaches based on learning or inference (as opposed to *ad hoc* approaches that do not require training data) [3]. Probabilistic matching, supervised learning, semi-supervised learning, and active learning
all fall into this first category. In 1959, Newcombe [8] successfully applied Bayesian inference to the realm of duplicate data record detection [3]. Fellegi and Sunter followed up this initial work by Newcombe by formalizing the problem space in the following way: given a comparison vector $x$ (containing data that may or may not be a match), as the input to a decision rule, an identification can be made whether comparison vector $x$ is assigned *match* or *no match* for the data in that vector [9]. A percentage of these comparison vectors could also fall into a *reject class* or category that contains record pairs for which it is not possible make any definite inference [9]. This category requires that a human expert perform a manual clerical review process, to effectively decide whether the pairs assigned to this reject category should instead be in one of the match/no match categories. It is important to note that this Bayesian inference technique is effective when there is no missing data and when all distributions are known, e.g., in the ideal case. However, this technique tends to degrade rapidly in real-world scenarios. For example, when the distributions of *match/no match* data points are unknown or when the typographical errors are high, the effectiveness of this technique degrades rapidly [3]. Du Bois points out that, in real-world data sets, not only are the distributions usually unknown, but also there is often missing data, which further degrades the effectiveness of this technique [10]. Du Bois suggests that Bayesian inference deduplication techniques completely ignore *non-matches* or *mismatches* that occur due to missing data. He does not advocate resolving these problematic data vectors in any way; instead, he recommends simply ignoring these data tuples [10].

The second group of deduplication techniques, referred to as *ad hoc* [3], consists of those techniques based on some kind of generic distance metric, and/or are based on
providing some kind of expert domain knowledge, thus avoiding the need for training data. As mentioned, this category of techniques does not require training, instead an appropriate matching threshold or thresholds, i.e., a minimum allowable similarity value for two elements to be considered a match, are chosen by a human domain expert [3].

In some situations, no training data is available, whether labeled or unlabeled. In such cases, learning based techniques or probabilistic models are not appropriate. This has led to a number of ad hoc approaches based on handcrafted comparison methods, for example, applying the combination of a distance metric, i.e., a tool/algorithm for comparing two strings, and a matching threshold, i.e., a minimum allowable similarity value for two elements to be considered a match, when detecting duplicates. Ad hoc deduplication, using a distance metric and threshold combination, is flexible enough to compare either an entire set of fields between two records or a subset/cross-section of the possible fields—both are viable approaches. Edit distance [11], tdf.idf [6], affine gap [12], q-grams [13], and Soundex [14] are all examples of distance metric algorithms that have proven useful for data deduplication. Fundamentally, these algorithms perform the same function, which is comparing two strings against each other to identify how similar, or dissimilar, the strings in question are. One implementation of distance metrics-based deduplication is to concatenate all of the fields of interest into one long field, and then use a distance metric, e.g., tdf.idf, q-grams, etc., to identify how similar these single field records are to each other [3]. Unfortunately, by concatenating multiple fields of data down into one long field, this approach can possibly ignore variations in the data that might be useful in identifying duplicate records.

As recently as 2005, Chaudhuri et al. showed that computationally setting distinct
and unique thresholds for real-world objects performed better than setting one, overall global threshold, when performing deduplication [15]. The identification of effective thresholds for ad hoc deduplication is a known problem, regardless of the entities represented in the database [3].

Rule-based deduplication is a specialized approach to generic distance metrics. A rule-based deduplication system leverages matching rules (or case-based reasoning) to detect duplicates. Either a human domain expert, or a probabilistic modeler like a Bayesian net, generates the rules [3, 8, 10]. These deduplication rules are typically comprised of standard distance metrics.

Matching

In data deduplication, the first step necessary is the identification of these matching or duplicate records. Whether through a principled approach, or through ad hoc methods like a rule-based system, the initial step is the same.

Several approaches exist for identifying these duplicate records. One of the most commonly used approaches in rule-based expert systems is to leverage one or more of the distance metric algorithms mentioned above. By intelligently combining a distance metric algorithm (or a combination of distance metrics) with some kind of threshold system, a rule-based deduplication system considers the similarity between a subset/cross-section of several records and compares the resulting similarities against each other. This process attempts to state, with some level of confidence, whether the records under consideration represent the same real-world, or at least unique, entity.

The process of identifying similar (duplicate) record data is the fundamental step necessary to deduplicate a record set and researchers continue to perform large amounts
of work to improve the algorithms used to compare two record fields against each other.

**Linking/Merging**

Once two or more records are identified as duplicate records, linking or merging these records into one representative and atomic record is the next step in a deduplication system. Several techniques exist for this process. One common approach includes merging the two duplicate records into one, new record, retaining distinct fields and intelligently merging competing fields [3].

**Blocking**

Unfortunately, the distance metric algorithms presented above do not focus on the efficiency of the deduplication process, nor do they attempt to address any concerns other than producing a quality comparison between two record fields, or set of record fields.

The time necessary to compare all of the records between two data sets, \( A \) and \( B \), is potentially \(|A| \cdot |B|\) (worst-case, when using something as simple as a nested loop to iterate over, and compare, every record in \( A \) with every record in \( B \)). Even if this record-comparison loop stops after finding only key differences, complete data deduplication can still require \(|A| \cdot |B|\) total comparisons (i.e., comparing all records, worst-case). The expense incurred by using such a technique grows according with the size of the data sets [3]. Further, the number of atomic data fields under consideration per record also compounds the expense of such a comparison technique. For example, consider a data set wherein each record contains ten fields. It is computationally more expensive as well as more complex to compare these ten-field records than to compare records containing only three fields.
Blocking or search space reduction addresses this cost and complexity by limiting the number of records for consideration to a specific subset of the data set [3]. Distance metric algorithms, such as Soundex [14], when applied to discriminatory data fields, such as last name [3], prove useful in creating these record subsets. Unfortunately, performing this blocking process can incorrectly exclude records that should be considered for deduplication (and therefore increase the false reject rate), thus limiting the efficacy of a deduplication system, regardless of the performance increase that blocking might introduce. Running the deduplication process in multiple passes, or runs, while blocking on different pieces of data, decreases the likelihood that this exclusion of the desired records will occur [3].

Classification

Once it identifies a set of records upon which to perform similarity metrics, the goal of a deduplication system is to generate a classification for each record under consideration. A binary classification of match or no match is the desired outcome. As mentioned previously, completely separable classification results (with no unknown labels) are not always possible; therefore, a clerical review category exists in some instances. Regardless, a binary classification decision of match or no match is most desirable when performing deduplication. Even so, a third category, e.g., possible match, could be helpful if we want to perform additional tests on the "possible" matches. This is often done to increase the speed of the initial classification.

A support vector machine (SVM) is one classifier that is useful and effective for assigning a category to data fields and/or records [16, 17]. In short, an SVM is a
supervised learning algorithm that leverages pre-labeled training data to learn how to categorize future data into one of two categories. In this case, we are interested in binary classification, but an SVM can also do multi-class classification and real-valued regression. The Library for Support Vector Machines (LIBSVM) is a well-accepted and easy to use open source SVM library [16, 17]. Specifically, LIBSVM provides vector classification support (C-support). Assigned one of two labels (or classifications), and several features or attributes (data points), training vectors inputted into LIBSVM are used to train the SVM to determine the classification of new feature vectors (often by solving an SMO problem defined by those feature vectors). When implemented as a linear classifier [16, 17], the SVM treats each input set of features as an n-dimensional data vector and learns to separate the input vectors with an n – 1 dimensional hyperplane. Through training, the SVM determines which hyperplane maximizes the distance between points in the distinct categories or classes. This hyperplane is the maximum-margin hyperplane.

Figures 2.1-2.2 present a graphical generalization of the determination of this maximum-margin hyperplane. Figure 2.1 shows the data points used for training. Note the visual classification of these points: X’s represent one class while O’s represent the second class. We train the SVM on these (labeled) data points to determine the aforementioned, and desirable, maximum-margin hyperplane that separates these two classes. Figure 2.2 shows the hyperplane for the representative training points from Figure 2.1.
Figure 2.1. This figure shows some generic and basic data points with no hyperplane separating the groups of data, where one category is made of Xs and the second category of Os.

Figure 2.2. This figure shows a representative maximum hyperplane for the categories presented in Figure 2.1. The dotted line separates the data points into two separate classes; the solid lines represent the margin between the two classes.
It is important to note that, as with most classifiers, the risk of over-fitting the SVM exists [16, 17]. Once over-fitted to its training examples, an SVM does not generalize well and thus performs poorly when presented with data that differs from the initial training data because the hyperplane used to separate the classified groups of data points is not maximized. The process of subdividing the training data into some predefined number of training sets and sequentially training the SVM on the subdivided training data—excluding one subset to act as the control/validation group (called cross-validation) can reduce this problem of over-fitting an SVM to its training data [16, 17]. Consider the over-fitted hyperplane presented in Figure 2.3. It does not separate the groups with the maximal hyperplane, thus the resulting classifications generated by this over-fitted SVM will be sub-optimal.

Linear classifiers cannot completely separate all data sets. When faced with data that is not linearly separable (such as the data shown in Figure 2.4), we can instead

![Figure 2.3. This figure shows a non-maximized hyperplane, when compared to Figure 2.2. This hyperplane is over-fitted to the problem space.](image-url)
employ an SVM with a nonlinear kernel. LIBSVM provides support for multiple kernels, including nonlinear kernels such as the radial basis function (RBF) kernel [16, 17]. Using a nonlinear kernel allows the classifier to project the original input data into a higher-order (nonlinear) space, in which it is hopefully easier to separate the data into distinct classes.

Previous Learning-based Deduplication Techniques

SVM classifiers and other learning-based techniques have been successfully applied to data deduplication. Bilenko et al. successfully used SVMlight (another commonly used SVM toolkit) in the supervised-learning deduplication problem space, showing that this technique outperforms other, more simplistic approaches, to identify matching or non-matching pairs [18]. In [18], Bilenko et al. employ an SVM to fuse the match score of all fields in the data record and output match/no match. This technique

![Figure 2.4](image)

Figure 2.4. This figure shows two classes of data that are not linearly separable.
performs well, but does not include any expert-created rules to determine which fields to compare. Instead, all fields are compared.

In addition, Christen showed that combining blocking with an SVM to perform record classification in a non rule-based environment works well [19]. Both of these previous techniques indicate that using an SVM classifier improves data deduplication, in general. However, neither of these techniques addresses the challenges of rule-based deduplication, namely, alleviating the need for manual rule tuning, and overcoming missing data.

Freely Extensible Biomedical Record Linkage

Privacy is a serious concern for real-world person-identification systems and therefore complete real-world identification data is often not publicly available [20]. The Freely Extensible Biomedical Record Linkage Toolkit, or FEBRL [21] is not only a well-known record linkage tool, but FEBRL is also a data set generator [21]. To generate data sets, FEBRL performs several steps. First, FEBRL generates random original records (based on real-world Australian census data), and then creates duplicate records from those original records, by introducing real-world field modifications, such as typographical errors, inserted characters, deleted characters, transposed characters and words, misspelled words, and missing fields. When generating duplicate records, FEBRL randomly selects a field within the record according to the selection probability for each field. After selecting a field, FEBRL randomly chooses a modification type (based on user-specified probabilities, real-world observed distributions, and a dictionary of possible misspellings) and applies that modification to the selected field. FEBRL
repeats this process until reaching the user-specified maximum number of modifications both per field and per record [20, 21].

As FEBRL generates biographical data, a schema based on such data is useful.

Global Justice XML Data Model

The global justice XML data model (GJXDM) provides a schema for the FEBRL-generated data. GJXDM is a standard schema for the storage and communication of biographic data (e.g., given name, surname, date of birth, and affiliations) and biometric data (e.g., fingerprint, face, and iris images) between U.S. government agencies [22]. This schema easily provides a framework for the data generated by FEBRL. Figure 2.5 provides a representation of the GJXDM schema.

Chapter Summary

The concepts and tools listed in this chapter set the framework for the techniques detailed and described in Chapters 3 and 4, as well as the additional experimentation laid out in Chapter 5.

Blocking provides a mechanism to keep the execution time reasonable, and

```
<PersonNameType>
  <PersonGivenName>Robert</PersonGivenName>
  <PersonSurName>Jones</PersonSurName>
  <PersonNameSoundexText>R163J520</PersonNameSoundexText>
</PersonNameType>
```

Figure 2.5. GJXDM XML fragment. GJXDM is a standard U.S. government schema containing person-identifying elements, such as surname, given name, and the Soundex code of both names.
several distance metric algorithms presented in this chapter provide a functional means of identifying duplicate and/or similar data record fields.

For our experiments, we generated several data sets containing real-world data. We used FEBRL to generate all of the training and test data, such that the GJXDM schema provides the framework for both the training and testing data.
CHAPTER 3

RULE-LEVEL FUSION FOR DATA DEDUPLICATION

Motivation

Rule-based deduplication utilizes expert domain knowledge to identify and remove duplicate data records. Achieving high accuracy in a rule-based deduplication system requires the creation of rules containing a good combination of discriminatory clues, i.e., salient comparison features. The rule-based approach to deduplication typically results in systems with high match accuracy [3]. However, the rules and thresholds used in these systems generally require many iterations of manual tuning by human experts. This process is not only time-consuming but is limited to the domain knowledge of a human expert, leaving the system imperfect, and resulting in ambiguous deduplication results that require clerical (i.e., manual) review. In these cases, a human analyst must perform a visual inspection to determine whether two data records represent an actual match [3]. This need for manual tuning reduces the efficacy of rule-based deduplication and its applicability to real-world data sets.

Even though manual tuning of the deduplication rules can reduce the need for this clerical review, by improving the efficacy and accuracy of the expert-defined rules, the benefits reaped by repetitive improvements to the rules become increasingly minute for the time trade-off. Currently, this cost/benefit ratio is not sufficiently addressed, as the state of the art approach in the development and tuning of rule-based deduplication systems, is as follows:
Generate matching rules, either using a supervised or semi-supervised system to automatically yield the matching rules via training data, or employing a human domain expert to produce the aforementioned matching rules.

Rely on a human expert to manually tune the generated matching rules over multiple iterations. Regardless of how the matching rules are generated, in practice, they must be manually tuned [3].

No adequate solution has previously existed for this manual tuning problem. The approach presented below eliminates the need for this extensive, manual tuning, while providing a high level of match accuracy.

**Overview of Technique**

Rule-level fusion is a novel technique for rule-based deduplication. This technique employs a hybrid of the two types of deduplication approaches previously mentioned, specifically principled and ad hoc. This technique utilizes match score-level fusion [1, 2] to tune the domain expert-defined rules (instead of manually tuning the rules) via supervised training and combines the individual rule results into one, overall, match-score. Key strengths of this solution include high accuracy, i.e., an order of magnitude better than non-tuned traditional rule-based deduplication, as well as automatic and effective tuning of the rules, thus alleviating the need for manual tuning of the deduplication rules and corresponding thresholds [3]. The proposed solution is principled but maximizes the strengths of rule-based deduplication, blending these differing approaches into a powerful, new approach.
This rule-level fusion deduplication technique is applicable to any rule-based deduplication system, as long as the system has access to the real-valued rule match-scores output by each rule. Here, we demonstrate our fused deduplication technique using a two-stage, rule-based deduplication system, originally presented in [23]. This rule-based deduplication system consists of two main pieces: (1) a set of computationally inexpensive primary rules, and (2) a set of more expensive secondary rules. Figure 3.1 describes, in pseudo code, the behavior of this two-stage rule-based deduplication technique without listing the specific deduplication rules. This figure is high-level, but instructive. Figure 3.2 provides a visual representation of the rule-level fusion technique described in this chapter, again without listing out the specific deduplication rules. The Experiment section below demonstrates the efficacy of this rule-based fusion technique on a set of specific deduplication rules.

**Rule Description**

**Primary Rules**

The primary rules perform blocking, i.e., search space reduction, for the secondary rules. As illustrated in Figures 3.1 and 3.2, the candidate set-building portion of the system steps through the list of primary rules in decreasing order of importance and applies the primary rules, one at a time, to the original set of records. The primary rules produce an initial set of probable matches by comparing the gallery of records to the probe (i.e., search) record. This set of probable matches is denoted as the primary candidate set. The system applies the secondary rules only to this primary candidate set. The deduplication rules behave in a short-circuited fashion: if the current primary rule
returns any candidates, the system immediately applies the secondary rules to those candidates. The system only applies subsequent primary rules if the secondary rules do not return any matches. Figure 3.3 provides a visual representation of this interaction.

```plaintext
//gallery is the list of all records in the database, p is the probe record
IdentifyDuplicateRecords(gallery, p) {

    create empty primaryCandidateSet
    for each record g in the gallery {
        for each primary rule {
            //test gallery record against rule-specific clues
            isCandidate = true
            for each clue in this primary rule {
                if(!clue.Satisfy(g, p)) {
                    isCandidate = false //failed to satisfy this clue
                    break
                }
            }
        }
    }

    //blocking
    if(isCandidate)
        add g to primaryCandidateSet

    create empty candidate/match-score list
    for each candidate record in the primaryCandidateSet {
        for each secondary rule {
            //use secondary rule to generate match-score
            matchScore = secondaryRule(candidate, p)
            add matchScore to the candidate/match-score list
        }

        matchValue = input candidate’s match-score into classifier (SVM)
        assign matchValue to candidate
    }
}
```

Figure 3.1. Pseudo code representation of rule-level fused deduplication technique.
Figure 3.2. Visual representation of the rule-level fusion matching machine.

Figure 3.3. The primary rules generate a candidate set, using a gallery of records as the source for the candidate set.
Each rule in this deduplication system is comprised of *clues*, i.e., salient comparison features. For the primary rules, each clue specifies (1) the field(s) to compare, (2) the number of characters to compare, and (3) the similarity identifying *clue function* (i.e., the type of comparison to perform). Figure 3.4 shows a generic representation of this relationship.

To keep the primary rules computationally inexpensive, the clues are limited to only a small set of salient record fields and typically compare only a substring of the actual field data. For example, a primary rule might compare only the first three characters of a field string, or simply verify that the field data is not null. Additionally, the primary rules tend to use inexpensive similarity functions, such as an exact string compare (as opposed to a more expensive similarity function like a complete edit distance). To generate the primary candidate set, the system compares the *probe record* to each record in the original gallery set, using the current primary rule, as described in Figure 3.3) If the gallery record satisfies all clues in the current primary rule, the system adds that record to the primary candidate list. Otherwise, the system effectively ignores that gallery record, and it is not included in the candidate set. In the experiments detailed below, a human, acting as a domain expert, specifies the primary rules prior to execution; the system loads these rules at run-time. Specific examples of the primary rules used in the experiments appear in the *Experiment* section below.
Secondary Rules

To produce the final deduplication results (the score set shown in Figure 3.2), this technique applies the secondary rules to the primary candidate set. The secondary rules return a list of matched, i.e., duplicate, records and their corresponding match scores. Similar to the primary rules, clues make up the body of each secondary rule. For the secondary rules, each clue specifies (1) the field(s) to compare, (2) the clue similarity function, e.g., Levenshtein edit distance [11], Soundex [14], string compare, etc., (3) the clue weight to add to the overall match score, and (4) the clue-level score threshold. Additionally, each secondary rule specifies a rule-level score threshold. As described in Figure 3.1, each clue in the secondary rule compares the probe record with the current candidate and calculates a match-score. If the clue match-score meets the clue-level threshold, the system adds the clue’s weight to the overall match-score.

Traditional (Non-Fused) Deduplication

In traditional non-fused rule-based deduplication, the matching algorithm applies the rule-level threshold after all of the clues execute within the secondary rule: if the summed match score, i.e., the sum of all of the match scores generated by each individual clue, meets or exceeds the rule-level threshold, the system adds the candidate record to the list of identified duplicates. Otherwise, if the summed match score fails to meet the rule-level threshold, the candidate record is not considered a duplicate. See Figure 3.5 for an example of this traditional rule-level threshold classification step.
//gallery is the list of all records in the database, p is the probe record
IdentifierDuplicateRecords( gallery, p ) {

    create empty primaryCandidateSet
    for each record g in the gallery {
        for each primary rule {
            // test gallery record against rule-specific clues
            isCandidate = true
            for each clue in this primary rule {
                if( clue.Satisfy( g, p ) ) {
                    isCandidate = false // failed to satisfy this clue
                    break
                }
            }
        }
    }

    // blocking
    if( isCandidate )
        add g to primaryCandidateSet

    }

    create matchList
    create noMatchList
    for each candidate record in the primaryCandidateSet {
        for each secondary rule {
            // use secondary rule to generate match-score
            matchScore = secondaryRule( candidate, p )
            classification = (matchScore > threshold) ? match : no match
            if( classification == match )
                add candidate to matchList
            else
                add candidate to noMatchList
        }
    }

Figure 3.5. A high-level view of the (non-fused) rule-based deduplication system.
Rule-Level Fused Deduplication

In the new technique presented in this chapter, the secondary rules operate in the same manner as in traditional non-fused deduplication: the clues contained in the secondary rules execute, allowing each secondary rule to produce a match-score at the rule-level by summing the match-scores generated by each clue within the rule. However, our fused deduplication technique does not rely on any manually tuned rule-level thresholds after calculating the rule match score. Instead, this technique ignores the original rule-level thresholds, and instead makes use of an appropriately trained SVM classifier to fuse, i.e., combine, the rule-level match scores and determine whether this candidate record is in fact a duplicate of the given probe record. Figure 3.6 shows this association between the secondary rules and the SVM classifier.

Figure 3.6. A generic example showing the relationship between rules, match-scores and an appropriately trained SVM classifier, when performing rule-level fusion.
Detailed Description

**Probe Record**

Let $r_0$, be the *probe record*. This probe record, or *data vector*, contains the original identifying features that the system considers when identifying whether other, distinct records contained in the gallery, are actually duplicates of $r_0$. In this case, the identifying features are the salient clues, as identified by a domain expert. The gallery of records to match against the probe record $r_0$ exist in a database of some kind, whether a relational database or in a flat file, it is only required that the records be distinct and separable.

**Filters**

Let $FS = \{f_1, f_2, ..., f_n\}$ be a filter set, where $f_i$ is an individual filter and functions as a Boolean expression over a database record’s scheme. Specifically, filters consider a subset of the fields contained in a database record and identify the similarity between two records according to individual and distinct field comparisons. If the similarity between the records is within some predefined range, as determined by the human expert-created threshold, a Boolean value of *true* is output. If not, the filter returns a Boolean value of *false*. For example, consider the filter represented in Figure 3.7. This filter compares two text fields against each other, resulting in a Boolean response of *true* if the first three characters of the strings are the same.
Primary Rules

Let $D = \{d_1, d_2, ..., d_n\}$ be the database under consideration, where $d_i$ is an individual record within the database and $|D|$ is the size of the database. It is possible that database $D$ contains hundreds, if not thousands, of records among which some should be considered for deduplication, whereas others should not. **Blocking**, i.e., **functional prescreening**, helps to minimize the data set under consideration by identifying which records, $d_i$, are possible matching candidates and assigns these records to a **candidate set**, $CS$. Hence, $CS = \{r_1, r_2, ..., r_n\}$ where $r_i$ is a deduplication candidate and $CS$ is a subset of $D$, denoted as $CS \subseteq D$, where $CS \ll D$.

This blocking is performed by generating a filter set, $FS$, and associating a **precedence**, or order of execution, with each filter, $f_i$, contained in $FS$. This $(f_i, Precedence)$ pairing constitutes a **primary rule** (PR), and a list of these primary rules, ordered by precedence, constitutes a **primary rule set** (PRS). The precedence, or order of execution, associated with a filter, is high for a more **strict** rule, or low for a more **lenient** rule. A PR is considered strict if the fields being considered are based on **strongly identifying** fields, whereas a lenient primary rule is based on more **ambiguous** fields which is less identifying. A strongly identifying field is one that yields very few incorrectly identified duplicates. For example, if two records contain the same value for

```c
Filter ( text_field_1, text_field_2 ) {
    return ( text_field_1 == text_field_2 first 3 chars );
}
```

Figure 3.7. A basic filter.
a field such as passportID, those records are likely to represent a duplicate. An ambiguous field is one that does not separate records as cleanly. For example, if two records contain the same surname, those records are not as likely to represent a duplicate.

If a PR identifies any possible candidate records, these records are added to the CS. After identifying the possible candidates, and assigning those records to the CS, the CS is returned for further processing, and no further PRs are immediately processed; thus, the PRS executes in a short-circuited fashion. If no CS is generated by executing the previous PR, or if no matches were found in the generated CS, the next PR in the PRS is executed, and any candidates found are returned and passed on for matching. Figure 3.8 shows an example of a generic primary rule.

**Clue Functions**

A clue function, CF, takes two parameters as inputs. The first parameter, \( r_0 \), is the probe record. As previously stated, this probe record contains the identifying features that the system is trying to match against the gallery. The second parameter, \( r_1 \), is a

```java
PrimaryRuleExample( ) {
    Create empty CS
    For each database record \( d_i \) in \( D \) {
        if (filter \( i \) return true) {
            add \( d_i \) to the CS
        }
    }
    return
}
```

Figure 3.8. A basic Primary Rule. This basic rule contains a single filter that creates the primary candidate set from the original gallery of records.
candidate record from a CS generated via the primary rules. A CF compares these two records against each other to measure the similarity between the records. This similarity is determined by comparing specific fields contained in the data records \( r_0 \) and \( r_1 \).

Soundex [14], the Levenshtein edit distance algorithm [11], tdf.idf [6], and token-based comparisons [24] are well-known techniques used to identify the similarity between two text strings. Figure 3.9 presents an example that uses Soundex to identify the similarity between two text fields (contained in records \( r_0 \) and \( r_1 \)).

**Clues**

Those fields containing discriminatory data, or groups of fields which prove discriminatory when grouped together, are used within a CF to identify the similarity between two records. If the similarity between two records is within some allowable edit distance, i.e., the allowed difference or similarity between the two records, the CF results in a Boolean value of true. Otherwise, a value of false is returned.

A clue, \( C \), associates a CF with a clue weight, \( CW \), creating a pair of the form \( C = \{CF, CW\} \), where \( CF \) is a clue function, and \( CW \) is a numeric weight assigned to that clue function. If the \( CF \) within a clue results in a Boolean value of true, the weight \( CW \) for that clue is returned. See Figure 3.10 for an example clue.

**Secondary Rules**

A secondary rule, defined as a set of user-associated clues, is written as \( SR = \{C_1, C_2, \ldots, C_n\} \), where \( C_i \) is an individual clue. These clues are executed sequentially where each clue returns either zero or the numeric clue weight, \( CW \), if the clue’s
Figure 3.9. A clue function that uses Soundex to determine the similarity between two text fields.
allowable edit distance is met. Once all of the clues used by a SR have returned their results, the accumulated resultant value is used as the match score, $MS$, for the entire rule. See Figure 3.11 for an example of a generic secondary rule.

In traditional rule-based deduplication systems, once this $MS$ is generated it is compared against the secondary rule’s rule threshold, which is the minimum value that the match score must reach for the two records, $r_0$ and $r_1$, to be considered the same entity. If the match score is not greater than or equal to the rule threshold, a zero is returned as the output for the current secondary rule, as shown in Figure 3.5. However, as shown previously in Figure 3.1, the proposed rule-level fused deduplication technique does not consider the rule threshold. Instead, it leverages rule-level fusion to identify match/no match. Thus, when performing rule-level fusion, the $MS$ is simply normalized to a value between 0 and 1 by dividing by the maximum sum of clue weights in the current rule, and is returned to be used as input for an SVM classifier.

```
ClueExample(r_0, r_1) {
    CW is set to some value
    Set the allowable difference between records
    If (the similarity between $r_0$ and $r_1$ is within the allowable difference{
        return $CW$;
    }else{
        return 0;
    }
}
```

Figure 3.10. This basic clue example shows the generic functionality of a clue used by a rule.
Support Vector Machine

An appropriately trained Support Vector Machine (SVM) takes the scaled match score values from each of the secondary rules as inputs, in the format of $(MS_1, MS_2, \ldots, MS_n)$, and performs match score-level fusion [1, 2]. This in turn yields one overall classification decision as follows.

$$RL - SVM(MS_1, MS_2, \ldots, MS_n) = \text{overall classification},$$

where $RL$ identifies this type of SVM as a rule level SVM, meaning that the $RL - SVM$ is applied to the match scores that are generated by the individual rules. An $RL - SVM$ is, therefore, an SVM that takes all scaled rule-level match-scores as inputs, regardless of value. Training this SVM to fuse the inputted match scores is a vital and necessary step of this technique. Without appropriate training, there is no confidence in the SVM's outputted value.

Rule-Level Matching Machine

Let $RL - MM = \{\{PR_1, \ldots, PR_n\}, \{SR_1, \ldots, SR_n\}, RL - SVM\}$, where $RL - MM$ is the overall rule-level fused deduplication technique described above, containing a set of primary rules, a set of secondary rules, and an appropriately trained SVM of type \[SecondaryRule_Example (r_0, r_i) \{
    \text{Initialize } MS \text{ to zero}
    \text{Set the total possible score for rule}
    \text{ } MS += \text{Clue1}(r_0, r_i);
    \text{ } MS += \text{Clue2}(r_0, r_i);
    \text{ } MS += \text{Clue3}(r_0, r_i);
    \text{ } MS += \text{Clue4}(r_0, r_i);
    \text{return } MS/ \text{ possible score; } // \text{normalized score}
}\]
Rationale for Training the SVM

We implemented the rule-level fusion SVM with LIBSVM [16, 17]. LIBSVM is an open-source library that supports SVMs for both classification and regression, and provides multiple types of kernels. In order to perform rule-level fusion, we must first train the rule-level fusion SVM. We train the SVM on data generated by FEBRL. FEBRL is a standard data set generation tool, based on real-world identification data [20, 21]. For a more complete description of FEBRL and the specific data sets used in our experiments, see the Data Sets section.

Steps for Training the SVM

Training the rule-level fusion SVM involves the following steps: (1) generating a set of training examples, based on the current set of deduplication rules and FEBRL-generated data set; (2) selecting a subset of those training examples; (3) determining the SVM parameters that produce the highest cross-validation accuracy; and (4) training the SVM to classify the given set of training examples, i.e., to solve the quadratic programming optimization problem described by the selected training examples. We discuss these training steps in more detail in the following sections.

Generating the Training Examples

To train the rule-level fusion SVM, we first create appropriate training examples. We generate the training examples by applying the current set of deduplication rules to the current FEBRL-generated data set. Thus, the training examples represent both the
current set of expert-created deduplication rules and the current FEBRL-generated data set. To generate the SVM training examples, we simply pass the current set of probe and gallery records into the original (non-fused) deduplication system, as described in the *Rule Description* section and shown in Figure 3.5. Note that we train the SVM on one FEBRL-generated data set and test on another data set, as discussed in the *Experiments* section [16, 17].

First, we create *single rule* match score examples, by (1) separately applying the individual secondary rules to the set of FEBRL-generated training records, and (2) outputting the individual rule-level match score, a rule label indicating which individual deduplication rule was used to calculate this match score, and whether or not this probe-gallery pair is an actual match (based on the FEBRL record ID of the probe and gallery records). We scale the single rule match scores to be in the range $[0, 1]$, where a value of 1 represents a perfect match [16, 17]. The scaled match score represents the percent matched between the probe and gallery records—as shown in Figure 3.11. We normalize the rule-level scores by dividing by the maximum possible match score, as defined by the current rule and its clue weights. The maximum possible rule-level match score is the sum of the current rule’s clue weights. When training an SVM, it is important to scale all of the input features so that they have a common range [16, 17]. This keeps any one input feature from dominating all other input features, simply due to scale.

Next, we concatenate the *single rule* match score examples of the same class to create *multi-rule* examples by taking the individual rule match scores (with the corresponding rule label and *match/no match* label) and then concatenating the individual rule scores, as shown in Figure 3.12. We generate training examples for the rule-level
SVM classifier, by applying the current set of deduplication rules to the current FEBRL-generated data set.

Finally, we label the multi-rule training example, based on the classification label of the individual rule-level match scores: when creating multi-rule training examples, we concatenate match scores of the same class, i.e., \textit{match}/\textit{no match}, so if all of the individual rule scores represent a match, then the combined example also represents a match.

When training the SVM for rule-level fusion, we do not limit the training examples to only those created from records with no missing field data. Instead, we create training examples from every record in the original data set, regardless of the amount of noise that it contains: we apply the current set of non-fused deduplication rules to every record in the original data set, comparing the probe records against the gallery records, and then output the rule-level match scores.

\textbf{Selecting the Training Examples}

After creating all of the multi-rule training examples for the current set of deduplication rules and current FEBRL-generated data set, we select a subset of examples on which to train the rule-level fusion SVM. We select the training examples randomly with replacement. We do not explicitly limit the selection of training examples to only unique examples, and therefore it is possible that the same training example is
selected more than once [16, 17]. As long as the training examples are selected in a sufficiently random manner, the SVM should generalize appropriately for the given training set [16, 17]. We do not want to over-fit the SVM to the training data; if over-fitted, the SVM will fail to generalize and will only be able to classify examples that it has already seen. Instead, we want new unseen (testing) examples to be classified correctly.

We select up to a user-specified number of SVM training examples: in our experiments, we typically used anywhere between 100 and 500 training examples. To account for the variability in example selection, we perform ten runs of training and keep the trained SVM with the highest cross-validation accuracy over all training runs. (We describe cross-validation accuracy in the next section.) See the Empirical Results section for a discussion of the specific number of training examples used in each experiment.

**Selecting the SVM Parameters**

In our experiments, we implement the rule-level fusion SVM with a radial basis function (RBF) kernel: $e^{-\gamma d^2}$. Using an RBF kernel requires that we set the value of the kernel weight parameter $\gamma$. In an RBF kernel, $\gamma$ represents the weight applied to the distance between the support vector, $u$, and the example, $v$. Additionally, a classification SVM, regardless of the type of kernel used, requires that we set the value of the constraints-violation cost, $C$, where $C$ represents the cost of violating the constraints of the SVM’s quadratic programming optimization problem, e.g., classifying an example with the wrong class label.

When training the SVM, we choose the appropriate $\gamma$-value and constraints-
violation cost, $C$, at run-time by performing $k$-fold (stratified) cross-validation on the selected set of training examples. We perform a grid search to find these parameter values, letting $\gamma$ be in the range $[2^{-15}, 2^3]$ with a step size of 4, and letting $C$ be in the range $[1, 2^{10}]$ with a step size of 4 [16, 17]. This parameter grid search contains the following steps:

1. Break the selected set of training examples up into five randomly chosen same-sized folds, i.e., subsets of training examples. These folds are created by randomly choosing examples from the set of selected training examples and assigning them to a fold.

2. Train the SVM on four of the five folds of selected training examples and hold out the fifth fold. We input the four folds of examples into the SVM and train it to solve the corresponding quadratic programming optimization problem, via sequential minimal optimization (SMO) [16, 17].

3. “Test” the SVM on the remaining held-out fold of selected training examples. Here, we let the SVM classify the unseen subset of training examples, i.e., the SVM predicts the label of each new example. Then we calculate and record the training cross-validation accuracy of the SVM, indicating the percentage of the new examples that the SVM classified correctly. Note that this cross-validation accuracy measures the sensitivity of the SVM, by focusing on the percentage of correctly classified examples.

4. Move to the next value of $\gamma$ and $C$, and repeat steps 1 through 3.

We keep the parameter values for $\gamma$ and $C$ that yield the highest cross-validation accuracy. After determining the best parameter values using this grid search, we then re-
train the SVM on the entire set of selected training examples (i.e., all five folds of training examples) [16, 17].

High training accuracy does not necessarily guarantee high testing accuracy on an entirely new set of test examples. However, keeping the parameters that produce the best training accuracy does indicate that the trained SVM has generalized appropriately for the selected set of training examples. As described in the Empirical Results section, the trained rule-level fusion SVM produces good deduplication results on new sets of FEBRL-generated testing examples and has therefore generalized properly.

Rule-Level SVM Fusion

We train the SVM to fuse the single-rule match scores and output one overall classification decision (for the entire deduplication system):

$$RL - SVM: R_1 \times R_2 \times \ldots \times R_n \rightarrow \{match/no match\}$$

where $R_1$ is the real-valued match score for secondary rule 1 (for the current probe-gallery record pair) and the current deduplication system contains some number of rules (represented by $n$).

Experiment

Data Sets

Privacy is a serious concern for real-world person-identification systems and therefore complete real-world identification data is often not publicly available [20]. For our experiments, we used FEBRL [21], a well-known record linkage and data set
generation tool, to generate data sets that conform to the GJXDM schema [22]. See Figure 2.5 for an example of the GJXDM schema.

To test our fused deduplication technique, we used FEBRL to generate several training and testing data sets. FEBRL generates data in the following manner. First, FEBRL generates random original records based on real-world Australian census data, and then creates duplicate records from those original records, by introducing real-world field modifications, i.e., errors or noise. To generate duplicate records from the original records, FEBRL randomly selects a field within the record according to the selection probability for each field. After selecting a field, FEBRL randomly chooses a modification type based on user-specified probabilities, real-world observed distributions, and a dictionary of possible misspellings and applies that modification to the selected field. FEBRL repeats this process until reaching the user-specified maximum number of modifications both per field and per record [20, 21]. FEBRL allows the user to specify the maximum number of modifications per field and per record at the command-line; this makes it easy to generate multiple FEBRL data sets using the same parameters.

To demonstrate the efficacy of our rule-level fused deduplication technique, we generate FEBRL data sets with increasing amounts of noise, by varying both the maximum number of modifications per field and the maximum number of modifications per entire record. As shown in Table 3.1, as the number of allowed modifications, i.e., noise increases, the average data set similarity tends to decrease—the duplicate records contain more noise and are therefore less similar to the original records. We measure the average similarity of the FEBRL-generated data set by calculating the average edit distance between each original FEBRL record and its corresponding duplicate records.
When calculating the average data set similarity (highlighted in the third column of Table 3.1), we consider all fields in each record. We flatten each record into a concatenated string of fields. We compare each original record against only the duplicates for that record, using the FEBRL record ID field to determine the corresponding original and duplicate records, and average this similarity calculation over the entire data set.

FEBRL introduces real-world modifications into the original records, including removing data (i.e., missing field data). The Percentage Missing column shown in Table 3.1 indicates the average percentage of all missing fields in the data set (regardless of which specific fields are missing).

Table 3.1. FEBRL-Generated Data Sets.

<table>
<thead>
<tr>
<th>Modifications per Record</th>
<th>Modifications per Field</th>
<th>Data Set Similarity</th>
<th>Percentage Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>89.77%</td>
<td>6.45%</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>72.00%</td>
<td>14.03%</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>52.81%</td>
<td>17.10%</td>
</tr>
</tbody>
</table>

For our experiments, we generated data sets ranging from 89.77% similarity to 52.81% similarity. We specifically generated data sets with an increasing amount of noise, in order to demonstrate that our fused deduplication technique performs well even on imperfect data sets. Each row of Table 3.1 represents five data sets generated with the same FEBRL parameters, i.e., the number of modifications per field and per record. Each data set described in Table 3.1 contains 1000 probe records and 1500 gallery records. We averaged the Data Set Similarity and Percentage Missing over these five data sets. The average values shown in Table 3.1 are consistent over all five runs. For
each data set, the standard deviation is less than 0.5%.

Method

As detailed above, the two stages of an $RL - MM$ are applied in a waterfall fashion, and are comprised of: (1) a set of computationally inexpensive primary rules, or $PRS$, and (2) a set of more expensive secondary rules ($SR$). We now describe the specific $PRS$ and the set of $SR$s used in this experiment.

Primary Rules

Figure 3.13 illustrates the two-stage deduplication method made up of primary rules and secondary rules, as detailed in the previous sections. For the experiments presented here, we created four primary rules, summarized in Figures 3.14-3.17. Primary rule 1 is the strictest and most important primary rule and is, therefore, applied first, with the rest of the primary rules applied in decreasing order of importance.

As described previously, the primary rules shown in Figures 3.14-3.17 perform blocking on the original set of gallery records. Blocking can increase the false reject rate (FRR) of a deduplication system, thus, constraining the search space increases the possibility of skipping an actual match. In order to reduce the expected FRR, Primary rule 4, the last rule to be employed, is a distinctly lenient rule (summarized in Figure 3.17) that utilizes Soundex [14] to phonetically compare proper names.
TwoStageDeduplication ($p$) 

```java
primaryCandidateSet = PrimaryRule1($p$)
if (primaryCandidateSet is not empty) {
    matches = SecondaryRules($p$, primaryCandidateSet)
    if (matches.bestScore > 0)
        return matches
}

primaryCandidateSet = PrimaryRule2($p$)
if (primaryCandidateSet is not empty) {
    matches = SecondaryRules($p$, primaryCandidateSet)
    if (matches.bestScore > 0)
        return matches
}
...
```

Figure 3.13. Two-stage, rule-based deduplication.

```
PrimaryRule1($p$) 

// is a probe record
create empty candidateList

for each $g$ in the gallery {
    if ($p$.passportID == $g$.passportID first 3 chars)
        if ($p$.passportCountry == $g$.passportCountry first 3 chars)
            if ($p$.visaID == $g$.visaID first 3 chars)
                if ($p$.visaType == $g$.visaType first 3 chars)
                    add $g$ to candidateList
}

Return candidateList
```

Figure 3.14. Primary Rule 1. Primary rule 1 is the strictest of the four primary rules, and relies on strongly identifying fields such as passportID.
Figure 3.15. Primary Rule 2. This more lenient rule is applied only if the first primary rule does not return any candidates.

```java
PrimaryRule2 (p) {  // p is a probe record
    create empty candidateList

    for each g in the gallery {
        if (p.passportID == g.passportID first 3 chars)
            if (p.gender == g.gender)
                if (p.nationality == g.nationality first 3 chars)
                    if (p.dateOfBirth exists)
                        if (g.dateOfBirth exists)
                            add g to candidateList
    }

    Return candidateList
}
```

Figure 3.16. Primary Rule 3. This rule is even more lenient than the previous primary rules: for most fields, it only requires a match in the first character.

```java
PrimaryRule3 (p) {  // p is a probe record
    create empty candidateList

    for each g in the gallery {
        if (p.gender == g.gender)
            if (p.nationality == g.nationality first 3 chars)
                if (p.surname == g.surname first char)
                    if (p.givenName == g.givenName first char)
                        if (p.orgName == g.orgName first char)
                            add g to candidateList
    }

    Return candidateList
}
```
Secondary Rules

The proposed secondary rules shown in Figures 3.18-3.20 produce the final deduplication results. The secondary rules calculate the match score for each probe-candidate pair and determine if that pair represents a duplicate.

As described in Figures 3.18-3.20, the secondary rules utilize identifying fields, or clues, such as passportID. When creating these rules, we weighted each clue by the discriminatory power of its underlying field(s). For example, as shown in Figure 3.18, we gave passportID a higher weight than passportCountry. This indicates that passportID, which represents the specific passport number, is a stronger identifier than the passport's issuing country. In other words, while many passports are issued by the same country, no two passports from the same country should share the same passport number.

Note the initial rule-level thresholds for the secondary rules, shown in Figures 3.18-3.20. We based these initial rule-level thresholds on the discriminatory power of the

```c
PrimaryRule4 ( p ) { // p is a probe record
    create empty candidateList

    for each g in the gallery {
        if( SoundexDistance( p.surname, g.surname ) < 0.75 ) {
            if( SoundexDistance( p.givenName, g.givenName ) < 0.75 )
                add g to candidateList
        }
    }

    Return candidateList
}
```

Figure 3.17. Primary Rule 4. Primary rule 4 is the most lenient of the four primary rules.
Figure 3.18. Secondary Rule 1. The secondary rules calculate the record match scores. This rule calculates the Levenshtein edit distance of identifying fields, or clues, such as passportID.

Listing 3.18: SecondaryRule1 (p, c) { //p is a probe, c is a candidate match
    matchScore = 0
    if (EditDistance(p.passportID, c.passportID) ≤ 1)
        matchScore += 10 // clue weight
    if (EditDistance(p.passportCountry, c.passportCountry) ≤ 1)
        matchScore += 5
    if (EditDistance(p.visaID, c.visaID) ≤ 1)
        matchScore += 8
    if (EditDistance(p.visaType, c.visaType) ≤ 1)
        matchScore += 2
    if (matchScore ≥ 20) //rule-level threshold
        return (matchScore/25)
    else
        return 0
}

Figure 3.19. Secondary Rule 2. This rule utilizes a higher number of clues containing less strongly identifying fields than secondary rule 1.

Listing 3.19: SecondaryRule2 (p, c) { //p is a probe, c is a candidate match
    matchScore = 0
    if (EditDistance(p.passportID, c.passportID) ≤ 1)
        matchScore += 8 // clue weight
    if (EditDistance(p.gender, c.gender) ≤ 1)
        matchScore += 3
    if (EditDistance(p.surname, c.surname) ≤ 1)
        matchScore += 6
    if (EditDistance(p.nationality, c.nationality) ≤ 1)
        matchScore += 3
    if (EditDistance(p.dateOfBirth, c.dateOfBirth) ≤ 1)
        matchScore += 3
    if (matchScore) //rule-level threshold
        return (matchScore/23)
    else
        return 0
}
SecondaryRule3 \((p, c)\) \{ // p is a probe, c is a candidate match

\[\text{matchScore} = 0\]

if (\(p.gender == c.gender\))
    \[\text{matchScore} += 3\] // clue weight
if (\(\text{EditDistance}(p.nationality, c.nationality) \leq 1\))
    \[\text{matchScore} += 3\]
if (\(\text{EditDistance}(p.surname, c.surname) \leq 2\))
    \[\text{matchScore} += 7\]
if (\(\text{EditDistance}(p.givenName, c.givenName) \leq 2\))
    \[\text{matchScore} += 5\]
if (\(\text{EditDistance}(p.orgName, c.orgName) \leq 2\))
    \[\text{matchScore} += 8\]

if (\(\text{matchScore}\)) //rule-level threshold
    return (\(\text{matchScore} / 26\))
else
    return 0
\}

Figure 3.20. Secondary Rule 3. This rule contains ambiguous fields, such as proper names, so this rule has slightly looser clue-level thresholds.

clues in each rule: if a rule contains ambiguous fields, we use a larger edit distance, i.e., clue-level threshold, and a looser overall rule-level threshold. As noted previously, rule-based deduplication systems are typically very accurate because they can be tuned to the current domain and data set, but this accuracy comes at the cost of manual tuning [3]. To demonstrate that our fused deduplication technique is not dependent on extensive manual tuning, we purposefully refrained from tuning the clue- and rule-level thresholds.

Rule Match-Score Level Fusion

Our fused deduplication technique proceeds in much the same manner as the non-fused rule-based deduplication system described above. However, our fused deduplication technique does not rely on manually tuned rule-level thresholds. Instead, we employ learning-based fusion [1, 2] to determine the appropriate rule-level thresholds.
[3], based on both the current data set and the discriminatory power of the individual rules (as described in Figure 3.1 and Figure 3.6). Our fused deduplication technique does not replace the domain expert. Instead, we utilize the original deduplication rules, and thus retain the expert domain knowledge used to create those rules. However, our fused deduplication technique reduces the manual tuning of those expert-created rules, using an appropriately trained SVM.

We trained the rule-level fusion SVM for the set of specific deduplication rules shown in Figures 3.14-3.20 in the same manner as described in the Training of SVM section, above. In our experiments, we trained the SVM on 300 training examples selected from a set of 1.5 million training examples, created by comparing 1000 probe records to 1500 gallery records. Empirically, using 300 training examples proved sufficient for training the rule-level fusion SVM for this set of deduplication rules; increasing the number of training examples did not noticeably improve the accuracy of the rule-level fusion SVM.

Testing Data

We used FEBRL to generate multiple data sets containing 1500 original and 1000 duplicate records [21], as described in Table 3.1. We trained the SVM on one of these FEBRL-generated data sets. To test the rule-level fusion SVM, we generated additional FEBRL data sets, using the same general parameters as the FEBRL-generated data set used to train the rule-level fusion SVM. We specifically trained on one data set and tested on a different data set, to avoid biasing the experimental results with the training data.

As described in the Data Sets section, FEBRL creates a random initial set of
records and then randomly selects the record and field to modify. This means that no two FEBRL-generated data sets will be the same. However, because FEBRL uses specific real-world probabilities both to select the field to modify and the type of modification to perform, multiple FEBRL data sets that are generated with the same real-world probabilities and the same parameters, including the number and distribution of duplicate records, have similar types and amounts of noise [20, 21]. Further, because we perform fusion on the output of the individual rules rather than on the specific contents of the record fields, we can train the rule-level fused deduplication system on one FEBRL-generated data set and test the system on another similar FEBRL-generated data set [1, 2].

**Testing the Rule-Level Fused Deduplication**

When testing the rule-level fusion SVM, we proceed much as in the training phase: we pass the current data set through the deduplication rules shown in Figures 3.14-3.20 and determine if any of the records are duplicates. As shown in Figure 3.6, the rule-level fusion SVM operates within the original deduplication system. We simply use the classification decision of the SVM to determine whether the current probe-candidate pair represents an actual match. The rest of the deduplication system, i.e., the primary and secondary rules and how they are applied to the probe and gallery records, operate in the same manner as the original non-fused deduplication.

The main difference between the training and testing phases is that when testing, we do not already know the label for the current vector of rule match scores. Instead, we apply the SVM classifier to predict the classification label of _match/no match_ for the current probe-candidate pair:
As described in Figure 3.1, when performing rule-level fusion we generate the single rule match scores, by separately applying each secondary rule to the current probe-candidate pair. See Figures 3.18-3.20 for a pseudo-code description of the secondary rules. We scale the single rule match scores to be in the range [0, 1] by dividing by the current rule’s maximum summed clue weights [16, 17]. Next, we concatenate the single rule match scores to create a multi-rule score vector as shown in Figure 3.21.

![Figure 3.21](image)

Figure 3.21. This figure displays the concatenation of single rule match-scores. The SVM predicts the classification (match/no match) of the probe-candidate pair.

Finally, we perform fused multi-rule deduplication, using an appropriately trained SVM. We apply the rule-level fusion SVM to the current multi-rule score vector, to determine its class label, i.e., match/no match. The SVM outputs one overall classification decision, indicating match/no match for the current probe-candidate pair. Unlike traditional rule-based deduplication, we do not apply the secondary rule-level thresholds when determining a match using rule-level fusion. Instead, we use the trained SVM to determine if the current scores represent a match.
Empirical Results

There are several terms that prove useful in quantifying and describing the experimental results; those terms include true negative (TN), false positives (FP), specificity, true positive (TP), false negative (FN), sensitivity, and accuracy [25].

A true negative (TN) is a correctly identified non-duplicate record.

A false positive (FP), is a non-matching record incorrectly identified as a match.

Specificity measures the ratio of correctly identified negatives (non-matching records) over the summation of all records identified as non-matches, whether correctly or incorrectly identified, such that:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

A true positive (TP) is a correctly identified matching record.

A false negative (FN) is a matching or duplicate record incorrectly identified as a non-match.

Sensitivity measures the ratio of correctly identified positives (matching or duplicate records) over the summation of all records identified as duplicates, whether this identification is correct or not. Such that:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

Figure 3.22 represents the common format in which we present the deduplication results, in terms of the above metrics. Additionally, accuracy is simply the number of correctly identified unique duplicates divided by the total number of unique duplicates in
the data set. This is a statement on the number of actual duplicates identified, rather than a statement on any ratio between positively and negatively identified records. Thus, accuracy reflects the sensitivity of the deduplication technique.

The experimental results shown below leverage the format presented in Figure 3.22. Sensitivity and specificity values effectively describe the efficacy of a deduplication system like ours because both measurements take into account not only the number of correctly identified records but also the incorrectly (or falsely) identified records. Even though both measurements prove important when describing experimental results such as ours, the nature of our training placed more emphasis on sensitivity rather than specificity. This emphasis is due to the cross-validation accuracy test used during the training of the SVM. Cross-validation accuracy based on the number of correctly classified labels indicates the number of true positives identified during a distinct training run of the SVM. Because of this type of validation, our trained SVM leans more towards sensitivity rather than specificity.

![Figure 3.22](image)

Figure 3.22. This figure shows the general outline of conditions to outcome for displaying the experimental results produced by an RL-MM.
For each record in the generated data set, FEBRL creates a record ID that indicates whether the record is an original or a duplicate; if a record is a duplicate, the FEBRL record ID also indicates which original record that duplicate corresponds to [20, 21]. This allows us to correlate records within the data set when calculating the accuracy of our deduplication technique, as well as when calculating the sensitivity and specificity of our technique.

Figures 3.23-3.25 show the experimental results obtained when performing deduplication on the FEBRL-generated data sets described in Table 3.1. The left-hand sides of these figures show the traditional non-fused deduplication results on one specific data set, and the right-hand sides of these figures show the rule-level fusion results on the same data set. The results shown in Figures 3.23-3.26 represent the total number of both correctly and incorrectly identified duplicates (for each technique). The cost of the incorrectly identified duplicates is the same as the benefit achieved by the correctly identified duplicates: we display the raw deduplication results without a specific penalty weight for the incorrectly identified records.

The fused deduplication results shown in Figures 3.23-3.26 were achieved with 300 training examples. All deduplication results shown in Figures 3.23-3.26 are averaged over five testing runs per FEBRL-generated data set presented in Table 3.1. We performed five runs of testing, using similar FEBRL-generated data created with the same FEBRL parameters wherein each of these data sets has the specified average data similarity, as described in Table 3.1. Each data set contains 1000 probe records and 1500 gallery records. When performing deduplication, the system compares each probe record against each candidate gallery record. This means that for each data set, the
Figure 3.23. The deduplication results for the first data set presented in Table 3.1 (with 89.77% data set similarity). This data set contains 3646 duplicates and 1495354 non-duplicates (for all 1000 probe and 1500 gallery records).

<table>
<thead>
<tr>
<th><strong>Traditional</strong></th>
<th><strong>Rule-Level Fusion</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive: 3325.2</td>
<td>False Positive: 0</td>
</tr>
<tr>
<td>False Negative: 320.8</td>
<td>True Negative: 1495354</td>
</tr>
<tr>
<td>Sensitivity 0.9120</td>
<td>Specifity 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Traditional</strong></th>
<th><strong>Rule-Level Fusion</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive: 3644</td>
<td>False Positive: 5</td>
</tr>
<tr>
<td>False Negative: 2</td>
<td>True Negative: 1495349</td>
</tr>
<tr>
<td>Sensitivity 0.9995</td>
<td>Specifity 0.9999</td>
</tr>
</tbody>
</table>

Figure 3.24. The deduplication results for the second data set presented in Table 3.1 (with 72.00% data set similarity). This data set contains 3644 duplicates and 1495356 non-duplicates (for all 1000 probe and 1500 gallery records).

<table>
<thead>
<tr>
<th><strong>Traditional</strong></th>
<th><strong>Rule-Level Fusion</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive: 1189</td>
<td>False Positive: 0</td>
</tr>
<tr>
<td>False Negative: 2455</td>
<td>True Negative: 1495356</td>
</tr>
<tr>
<td>Sensitivity 0.3263</td>
<td>Specifity 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Traditional</strong></th>
<th><strong>Rule-Level Fusion</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive: 2892.6</td>
<td>False Positive: 101.8</td>
</tr>
<tr>
<td>False Negative: 751.4</td>
<td>True Negative: 1495254.2</td>
</tr>
<tr>
<td>Sensitivity 0.7937</td>
<td>Specifity 0.9999</td>
</tr>
</tbody>
</table>
Figure 3.25. The deduplication results for the third data set presented in Table 3.1 (with 52.81% data set similarity). This data set contains an average total of 3608.6 duplicates and 1495391.3 non-duplicates (for all 1000 probe and 1500 gallery records).

deduplication systems performs up to 1.5 million total comparisons for all probes (1000 probe records compared against 1500 gallery records).

As represented in Figures 3.23-3.25, when the similarity is high between the probe records and the gallery records, the sensitivity and specificity differences between the two deduplication techniques is not dramatic. In Figure 3.23, the sensitivity values between the traditional technique and the \( RL - MM \) is only 0.912 versus 0.9995 (or 91.2% versus 99.95%). However, note that as the similarity degrades between the original data records and the FEBRL-generated duplicate data records, the difference in the sensitivity of the two techniques begins to drastically increase. When performing deduplication on the second data set wherein the average similarity was 72%, the traditional non-fused deduplication produced a sensitivity of 0.3263 versus a sensitivity value of 0.7937 with rule-level fusion (see Figure 3.24). Moreover, in the final data set, which was intentionally noisy and difficult, with a similarity of only 52.81%, the sensitivity of the two techniques differs even further, with the traditional deduplication
evaluating to 0.0229 and the $RL - MM$ evaluating to 0.5616.

One aspect of this increased dissimilarity between the efficacies of the two techniques is the decreased number of false negatives identified by each technique. Even though the $RL - MM$ still identified false negatives, the amount of false negatives is less than those identified in the traditional rule-based technique. Consider the first experiment, when similarity between data records is high (Figure 3.23), the $RL - MM$ only identified an average of two false negatives, whereas, on average, the traditional approach falsely identified 320.8 records as non-duplicates. The $RL - MM$ identified only 0.6% as many false negatives as the traditional technique. That ratio is not quite as high when the data similarity degrades, but the simple quantity of false negatives is still drastically reduced. In Figure 3.24, the difference is an average of 2455 to 751, and the final experiment (Figure 3.25) results in 3526.3 versus 1582.

Even though the number of false negatives increased with more troublesome data, as shown in Figures 3.24 and 3.25, the $RL - MM$ still managed to significantly outperform the traditional technique when identifying true positives. In Figure 3.23, the difference between the two techniques is this: 3325.2 true positives identified by the traditional deduplication versus 3644 true positives identified by the $RL - MM$. This increase in true positives is not drastic, but as seen in Figures 3.24 and 3.25, the $RL - MM$ identified more than twice as many true positives than the traditional technique identified with a more noisy data set (at 72% data set similarity). Furthermore, the $RL - MM$ identified 96% more true positives than the traditional technique at 52.81% data set similarity.

The traditional sensitivity values, shown in Figures 3.23-3.25, could be improved
by iterative manual tuning of the rules and their corresponding thresholds [3]. The original deduplication rules shown in Fig. 3.14-3.20 represent a good initial attempt by a domain expert. However, we purposefully did not manually finely tune these deduplication rules. Instead, we employed learning-based fusion to alleviate the necessity of manual rule tuning; the deduplication results shown in Figures 3.23-3.25 demonstrate that rule-based fusion reduces the need for manual rule tuning.

While the $RL - MM$ provided a higher sensitivity in our experiments than the traditional deduplication approach, the specificity for our experiments was lower due to the number of false positives identified by the $RL - MM$. A high sensitivity provides confidence that the $RL - MM$ correctly identified actual duplicate records, whereas a high specificity indicates that the non-duplicate records were correctly identified as non-matches. The $RL - MM$ performed noticeably better on the least noisy data set (see Figure 3.23), and understandably less well on the more noisy data (see Figures 3.24-3.25). However, even on the noisiest data set (Figure 3.25), the $RL - MM$ outperformed the sensitivity of the traditional non-fused deduplication, indicating that rule-level fusion overcomes the need for manual rule tuning.

When compared to the number of false positives identified by the traditional technique, the $RL - MM$ performed much worse, as the traditional technique consistently identified no false positives. This indicates that the original deduplication rules were likely too strict for the given data sets. The original rules appear to reject possible matches; hence, the traditional deduplication has a low false positive rate but a high false negative rate. Conversely, the fusion SVM proved far more sensitive due to the cross-validation training employed during the training phase. This type of training allowed the
SVM to learn smarter threshold levels than the initial domain expert-defined thresholds. Even though trained for the current rule set, the fusion SVM technique does not reject as many possible matches as the traditional deduplication technique. Additionally, the data sets shown in Figures 3.23-3.25 contain a large number of non-duplicates, i.e., true negatives. Therefore, even though the $RL - MM$ has a higher false acceptance rate than the traditional deduplication technique, the $RL - MM$ still has a reasonable specificity. Furthermore, even though using an SVM in the place of manual tuning caused the specificity to decrease, as the amount of noise in the data set increased, the rule-level fusion’s ability to correctly identify duplicate records increased dramatically.

Figure 3.26 shows an overview of the results presented in Figures 3.23-3.25. This figure presents the accuracy (sensitivity) of the two contrasting techniques for each data set detailed in Table 3.1, along with the percentage of missing data in each data set. As Figure 3.26 shows, rule-level fused deduplication produces higher sensitivity than the traditional non-fused deduplication, particularly as the amount of noise in the data set increases.

As Figure 3.26 presents, when the data quality decreases, the overall accuracy for each technique also degrades. However, for each data set, the $RL - MM$ outperformed the traditional rule-based technique, particularly as the amount of noise in the data set increases. For example, when the data-set similarity is at 89.77%, the accuracy of both techniques is over 90%. Even though both techniques perform well in terms of accuracy when data-set similarity is high, as this similarity drops and the amount of noise in the data set increases, the accuracy of the traditional matching technique drops much more drastically than when applying the $RL - MM$. For the second data set, with 72.00% data
Figure 3.26. Overall view of the sensitivity, data similarity and matching percentage achieved for each data set.

similarity, the difference in accuracy is at more than 40%, and in the third data set, where the data similarity is just over 50% and the missing data percentage is at 17.10%, the traditional technique only managed to identify 2.28% of the duplicate records accurately.

These results also show that the $RL-MM$ also suffers in the presence of missing data. Figure 3.26 shows that the accuracy dropped by over 20% between the first and second data sets, as well as another 23% when comparing the second data set with the third. Even though the similarity in the data set dropped between those data sets, the type of noise shifted away from transposed characters, white space, etc., and moved more towards missing entire fields within each duplicate record, thus the move from 6.45% missing data in the first set of experimental runs, all the way to 17.10% missing data in the final set of experiments. As more noise was introduced to the data set, the amount of
missing field data increased, likely due to the increased number of allowed modifications per field. The more modifications that are allowed, the more likely we will see all modifications types, including missing data.

The sensitivity values, as well as the accuracy values shown in Figures 3.23-3.26, for the traditional rule-based deduplication technique reflect the fact that we did not finely tune the set of rules we used. Instead, we demonstrated that our fused deduplication technique produced high average accuracy, even when fusing imperfect individual rules. These results imply that our fused deduplication technique alleviated the need for manual rule tuning. Reducing this manual tuning makes rule-based deduplication plausible for real-world data sets [3, 7, 20, 21].

The deduplication accuracy values shown in Figures 3.23-3.26 are specific to the set of deduplication rules employed. A different set of rules will yield different accuracy values for both fused and non-fused deduplication (as shown in Chapter 5). However, if the deduplication rules produce consistent match scores, and the fusion SVM is trained appropriately, the use of information fusion should increase the deduplication accuracy. In Chapter 4, we demonstrate our fused deduplication technique on four additional data sets. We demonstrate our fused deduplication technique on two additional sets of rules in Chapter 5.

**Discussion of Strengths and Weaknesses**

The empirical results show that the original deduplication rules performed mildly well when executed in a traditional, non-fused deduplication system, relying on the human expert-defined initial rule thresholds. However, when using the same deduplication rules and ignoring the initial rule-level thresholds, an appropriately trained
SVM classifier can act as the decision maker, providing a system with an increased overall accuracy. This shows that using an SVM to provide the final classification step in rule-based deduplication is a powerful tool, and appears to remove the need for iterative manual tuning of the rules and their associated thresholds.

Even while leveraging an SVM classifier to remove the need for iterative manual tuning, rule-based deduplication systems remain heavily dependent on the efficacy of the original rules. If the primary or secondary rules are ineffective or badly designed, the efficacy of the system drops accordingly. For example, if the original deduplication rules output highly inconsistent match scores, the SVM can fail to learn an appropriate system-level threshold or the SVM can require significantly more training examples to effectively learn how to combine the individual rule-level match scores.

In cases wherein the distribution and probability of noisy data within the data set changes significantly, the deduplication system can require additional tuning. When using a fusion SVM, the SVM does not need to be trained from scratch when there is a significant shift in the record data; instead, the old SVM can be used as a starting point for online SVM training. Even if it is required to start from scratch and fully retrain the SVM, it is not a time-consuming process and is significantly less costly than manually tuning the matching rules and associated thresholds. In our experiments, we trained the rule-level fusion SVM in about one minute using a standard dual-core processor, even with multiple training runs.

It is important to note that the rule-level fusion technique described above is dependent on the existence of data in the training and testing data sets. If the individual fields in the data records are missing in a sufficient quantity, the SVM will either (1) not
be trained sufficiently or (2) not be able to provide an accurate classification due to a lack of salient data points in the test data. As shown in Figures 3.23-3.26, as the level of missing data increased, the returns gained by leveraging an SVM in the place of manually tuned thresholds decreased. This is a consequence of the \( RL-MM \) technique as a whole. An \( RL-MM \) fuses all of the rule-generated match-scores; therefore, it is trained to use all of them. When data is missing, the efficacy of these match-scores drops. Therefore, even though this technique still performs better than the non-tuned traditional rule-based deduplication, its effectiveness degrades significantly in the presence of missing data. In the next chapter, we present a technique for performing deduplication with missing field data.
CHAPTER 4

ROBUST DEDUPLICATION WITH MISSING RECORD DATA

Motivation

To achieve a high level of accuracy in rule-based deduplication, the deduplication system requires data values to exist in each record field under consideration. If no data exists for a given field, that field cannot contribute any explicit information to the deduplication rule. Thus, as the amount of missing data increases, the accuracy of the expert-created deduplication rules, and therefore of the entire rule-based deduplication system, tends to degrade. Unfortunately, missing data values, i.e., incomplete records, are common in real-world data sets [3, 6, 7, 10, 20, 21]. Thus, we would like to retain the benefits of rule-based deduplication systems, such as the knowledge of the domain expert, while maintaining good deduplication accuracy even when there is missing data. Additionally, we would like to continue leveraging the power of learning-based fusion to alleviate the need for manual tuning of the deduplication rules.

The novel learning-based fusion approach to rule-based deduplication presented in this chapter provides robust behavior against missing record data. Even though this technique is similar to the rule-level fusion technique detailed in Chapter 3 [23], the application of where, when, and upon what to perform fusion is different. In short, this more robust clue-level fusion technique performs deduplication by (1) calculating the individual match score ($MS$) for each clue within a deduplication rule, (2) selecting the Support Vector Machine (SVM) that best fits the existing clues, and (3) fusing, i.e., combining, the individual clue match-scores, using the selected best-fit SVM [4].
The fused deduplication technique detailed in Chapter 3 [23] is a rule-level fusion technique for biographic text deduplication. That rule-level fusion technique replaces manual rule tuning with learning-based fusion, thereby increasing the accuracy of a rule-based deduplication system without requiring iterative and potentially time-intensive manual tuning. The rule-level fusion technique uses a single SVM to perform fusion at the rule output-level, and as the empirical results in this chapter show, the rule-level fusion technique described in Chapter 3 is not robust against missing and partial record data.

In this chapter, we propose a fused deduplication technique that is robust against missing data. We show empirically that our fused deduplication technique achieves good deduplication results, even when portions of the record data are missing. We further demonstrate that our fused deduplication technique alleviates the need for manual rule tuning that is typically required for rule-based deduplication systems [3].

Overview of Technique

We perform fused rule-based matching on text records by (1) calculating the individual clue match-scores within each deduplication rule, (2) choosing the one SVM that best fits the existing clue data, and (3) fusing the individual clue match-scores (within each deduplication rule) to determine if the current record is a duplicate.

We demonstrate the benefits of our fused deduplication technique on records that comply with the GJXDM schema [22]. Government databases, such as state and national census databases, tend to be very large and frequently contain both inexact duplicates and
incomplete records, i.e., missing data [7, 20, 21]. Accurately deduplicating these databases despite the missing record data is a significant real-world problem.

Note that the rule-level matching machine, $RL - MM$, presented in Chapter 3 [23], differs from the fused deduplication technique presented in this chapter since the $RL - MM$ applies single-SVM fusion at the overall deduplication system level after each of the deduplication rules has been applied. Therefore, it does not account for missing data within a record. The technique presented in this chapter, however, uses best-fit SVM fusion [4] within each rule, i.e., at the clue level, to be robust against missing data.

As with the rule-level fusion technique described in Chapter 3, the clue-level best-fit SVM fusion described in this chapter uses a primary rule set (PRS), containing computationally inexpensive matching rules to perform blocking on the entire record gallery in an effort to generate a candidate set (CS). Once a CS containing potentially matching candidates is ready, it is passed off to the more expensive secondary rules for text field comparisons, as shown in Figure 3.5. However, that is where the similarities between the two techniques begin to end.

As with the previous technique, these secondary rules are composed of clues and clue functions. See Figure 4.1 for a generic example of this relationship.

Unlike rule-level fusion wherein each secondary rule is executed and the match-scores (MS) outputted by the secondary rules are then used as input into a single static SVM classifier, the fusion here is performed with different inputs, and it is performed at an earlier stage in the deduplication process. The clue-level fusion presented here
Figure 4.1. A generic rule containing clues and clue functions.

differs from rule-level fusion. It differs in the following ways: instead of one single, static SVM that takes the individual rule-level match-scores as input, this clue-level fusion technique applies one so-called best-fit SVM selected from a bank of prebuilt SVMs to the set of clues that comprise each secondary deduplication rule, using the match-scores generated at the clue-level as inputs. This bank of prebuilt SVMs acts as a library or list of available SVMs all trained and made available to the system so that at run time, the system can leverage the appropriately trained SVM, dependent on which clues return useful match scores. Figure 4.2 presents this new clue-level fused deduplication technique in a generic fashion.

Figure 4.2 shows a generic Secondary Rule (SR) containing a set of clues (denoted as $C_1$, $C_2$, and $C_3$). In this technique, the individual clues generate a similarity measurement, i.e., a clue-level match-score. In Chapter 3, these clue-level match-scores are summed up into one rule-level score for each $SR$. Here, however, these clue-level match-scores exist distinctly from each other and act as inputs into the appropriately trained SVM classifier. In Figure 4.2, these clue-level match-scores are marked as score $c_1$, score $c_2$, and score $c_3$. The system then scales those match-scores between $[0, 1]$ and makes these scores available as inputs for an SVM. Once the scaled match-scores are available, the system selects the appropriately trained SVM based on which clues have
data and applies the SVM to the clue match-scores, allowing the selected best-fit SVM to classify the current record as either a match or no match, where match indicates a duplicate record. Figure 4.3 provides a generic example of the best-fit SVMs generated from the clues in Figure 4.2.

As seen in Figure 4.3, we train one SVM for each possible (unique) combination of clues contained in each secondary rule. Note that each best-fit SVM considers two

---

**Figure 4.2.** A generic representation of the clue-level best-fit SVM deduplication.

**Figure 4.3.** We train a set of best-fit SVMs for the clues contained in Secondary Rule, SR (shown in Figure 4.2).
clues at a minimum. As there is no way to perform fusion on a single piece of data, we only train SVMs that fuse at least two distinct clues. Once trained, these SVMs are then stored and made available at runtime in descending order, wherein the first best-fit SVMs considered for fusion are those that perform classification on the highest number of clues.

Creating this bank of SVMs provides the system with the ability to perform fusion when one or more clues cannot return a useful match-score. The system ignores any missing clue data and fuses only the existing clue match-scores. For example, consider the case in which the first clue in Figure 4.2, $C_1$, attempts to determine the edit distance between two text fields, but unfortunately, the first text field is empty because it was missing in either the probe or gallery record. In this case, the match-score $MS$ outputted by the clue is zero. In rule-level fusion, that missing score would be included when generating the rule-level match-score, dropping the overall match-score. However, in this new clue-level fusion technique, an SVM exists that performs fusion using the other, non-empty fields (see $SVM_3$ in Figure 4.2, which ignores clue score $c1$). The clue-level fusion SVM fuses around the missing individual clue score and still takes advantage of the clue data that does exist. Rule-level fusion, i.e., fusion at the rule-output level, cannot fuse around individual missing clues because it operates at a higher level of system granularity and does not have access to the individual clue data.

In short, this new clue-level fused deduplication technique leverages primary rules to perform blocking, then uses the clues contained within the expert-defined secondary rules to generate clue-level match-scores on individual record fields. Once generated, the system selects the appropriately trained SVM from a bank of prebuilt best-fit SVMs, and
inputs the match-scores, resulting in a classification for the probe-candidate pair in question. Figure 4.4 provides a high-level graphical representation of this technique.
A more detailed and thorough discussion of this technique (including more specific examples), as well as more details on the creation and training of the best-fit SVMs, is contained in the Detailed Description section below.

## Detailed Description

### Clue Set

Recall that a secondary rule, $SR$, is comprised of a set of clues, such that $SR = \{C_1, C_2, ..., C_n\}$, where $C_i$ is an individual clue. This list of clues, or clue set $R$, makes up the functional body of the $SR$, allowing it to determine the similarity between two records containing text fields. For example, consider the set of clues, $R_1$:

$$R_1 = \{\text{passpportID, passportCountry, visaID, visaType}\}$$

### Best-Fit Support Vector Machines

For best-fit SVM fusion, we train one SVM for each unique combination of at least two clues. Figure 4.5 presents the distinct combinations of clues generated from $R_1$.

$$R^* = \{\{\text{passportID, passportCountry}\}, \{\text{passportID, visaID}\}, \{\text{passportID, visaType}\}, \{\text{passportCountry, visaID}\}, \{\text{passportCountry, visaType}\}, \{\text{visaID, visaType}\}, \{\text{passportID, passportCountry, visaID}\}, \{\text{passportID, passportCountry, visaType}\}, \{\text{passportID, visaID, visaType}\}, \{\text{passportCountry, visaID, visaType}\}, \{\text{passportID, passportCountry, visaID, visaType}\}\}$$

Figure 4.5. This figure shows distinct clue combinations created from one secondary rule's clue set.
Here, $R_1^*$ is the power set of the individual clues that comprise $R_1$ minus those sets with cardinality less than two (because fusion via an SVM requires at least two points of data). Let $s$ represent each distinct feature set, or unique subset of clues, identified in $R_1^*$.

Our best-fit fusion system trains up to $2^{|R_1^*|} - (|R_1^*| + 1)$ SVMs for $R_1$, where each SVM learns to fuse a specific feature set, $s$. This is the maximum number of SVMs learned by our fused deduplication system, per secondary rule.

The secondary rules are not necessarily disjoint, meaning that the clues or even the clue sets could overlap between the secondary rules. For example, passportID and visualID could be included in more than one expert-defined secondary rule. Therefore, training new SVMs only occurs for each unique feature set, i.e., combination of clues; if a subsequent rule contains feature set $s$, for which an appropriately trained SVM already exists, there is no need to train a new SVM as this technique shares the list of best-fit SVMs across all secondary rules. This reuse is possible because this best-fit SVM technique uses the clue match-scores without first applying the specific clue-level thresholds, unlike the traditional non-fused deduplication system described in Chapter 3, which uses not only the clue-level thresholds, but also the rule-level thresholds in identifying matching records.

Note that creating one SVM per unique subset of clues can result in many SVMs to train. If a deduplication system contains many deduplication rules and if those rules are disjoint, i.e., the rules do not contain any of the same clues, the total number of unique clue subsets can be large. Initially, the number of features to combine is constrained by the expert-created rules. Essentially the expert-created rules perform
**feature selection**: we train an SVM for each subset of clues in the expert-created rules, wherein the domain expert has already determined the salient clues by selecting and intelligently grouping together features of the text-field records that are useful for identifying matching data records \([1, 2, 3]\). For rules that are created and understood by a human domain expert, there is an implicit limit on the number of clues that will appear in each rule.

Despite this implied limitation, in the case wherein so many unique clue subsets exist that training all of the best-fit SVMs becomes impractical, one could perform additional pre-training feature selection to reduce the number of unique clues \([3, 10, 15, 19, 20, 21]\). In our experiments (see *Empirical Results*), the expert-created rules proved sufficient to deduplicate GJXDM records and contain few enough unique clues that no additional feature selection was required.

**Clue-Level Fusion**

As described in Chapter 3, traditional non-fused rule-based deduplication determines matching records by using the individual clue match scores and clue- and rule-level thresholds. If a record has no data for a particular clue, that clue does not contribute anything to the overall match score. Thus, as the amount of missing data increases, more and more of the clues are rendered useless, and the accuracy of the non-fused deduplication system decreases.

Instead, to maintain good deduplication accuracy, even with missing data, i.e., partial records, we apply clue-level best-fit SVM fusion (see Figure 4.2) to the clues that comprise each secondary rule:

- For the current secondary deduplication rule, \(SR\):
o Calculate the individual match score for each clue, using that clue’s similarity function, e.g., Levenshtein edit distance [11], Soundex [14], or other appropriate distance measure.

o Do not apply the clue-level threshold to the individual clue match-scores. Instead, if there are at least two clue match-scores to combine, fuse the existing clue match-scores:

- Select one fusion SVM that best matches the current set of clues, where $s \in R^*_S$. This technique selects the best-fit SVM based on the existence of clue data. If a record has no data for a specific clue, that clue and its corresponding match-score value of zero is not included in the fusion.

- Apply the selected best-fit fusion SVM to the individual clue match-scores, and generate one overall rule classification decision, indicating match/no match. The output of the best-fit fusion SVM indicates whether the current individual clue scores represent a duplicate record.

o In the case wherein there are less than two clue-level match-scores to combine, apply the original (non-fused) secondary rule. There will be less than two clue match-scores if the original deduplication rule only contains one clue, or if the current probe and gallery records are missing data for all but one clue. An SVM cannot perform information fusion on a single data point because there is nothing to fuse (combine). Thus, if there is only one clue match score, this technique simply uses the original non-fused
deduplication rule. Figure 4.6 provides a logical representation of this decision algorithm.

It is important to note that this best-fit SVM fusion technique differs from a traditional ensemble. Rather than applying every SVM to the clue match scores and then combining the output of all of the SVMs, by definition, this best-fit SVM approach only applies the one SVM that best fits the existing clue data.

For clarity, consider that the selected SVM contained in the bank of appropriately trained SVMs takes the scaled clue-level match-score values as inputs in the format of $(MS_1, MS_2, \ldots, MS_n)$, performing match score-level fusion [1, 2], resulting in one overall rule classification decision, such that $CL - SVM (MS_1, MS_2, \ldots, MS_n) = \text{overall rule classification}$. $CL$ identifies this type of SVM as a clue-level SVM.

**Clue-Level Matching Machine**

Let $CL - MM = \{(PR_1, \ldots, PR_n), \{SR_1, \ldots, SR_n\}, CL - SVM\}$, where $CL - MM$ is the overall clue-level fusion technique described above and contains a set of primary rules, a set of secondary rules, and a bank of appropriately trained SVMs of type $CL - SVM$.

Figure 4.6. Logic diagram representing clue-level fusion.
Rationale for Training the Bank of Best-Fit SVMs

We implemented each best-fit clue-level fusion SVM with LIBSVM [16, 17]. LIBSVM is a commonly used open-source library that supports SVMs for both classification and regression, and provides multiple types of kernels. In order to perform clue-level fusion, we must first train the bank of clue-level fusion SVMs. In our experiments, we performed clue-level fusion on data sets generated by FEBRL [20, 21].

Steps for Training the SVM

Training each SVM involves the following steps: (1) generating a set of training examples, based on the current set of deduplication rules and the current FEBRL-generated data set; (2) selecting a subset of those training examples; (3) determining the SVM parameters that produce the highest cross-validation accuracy; and (4) training the SVM to classify the given set of training examples, i.e., to solve the quadratic programming optimization problem described by the selected training examples.

Generating the Training Examples

To train the clue-level fusion SVMs, we first create appropriate training examples. We generate these training examples by applying the current set of deduplication rules to the current FEBRL-generated data set. To generate the clue-level fusion training examples, we simply pass the current set of probe and gallery records into the non-fused deduplication system; we output the clue-level match scores generated by applying the current set of deduplication rules to the current FEBRL-generated data set.

When creating the clue-level fusion examples, we ignore the initial clue-level thresholds used in the original non-fused deduplication system. As previously discussed,
both here and in Chapter 3, the clues used in traditional non-fused rule-based
deduplication use these thresholds to determine the classification of a record, as shown in
Figure 4.7.

The $CL-MM$, however, uses SVM-based fusion to specify this classification,
i.e., $match/no\ match$, at the clue level. As shown in Figure 4.8, when performing clue-
level fusion, the secondary rules do not aggregate the clue-level outputs into one overall
match-score and compare that clue score against some threshold. Instead, the secondary

```
ApplySecondaryRules (p, primaryCandidateSet) { //p is a probe record
  create empty matchList
  create empty noMatchList

  for each secondary rule SR {
    for each candidate in primaryCandidateSet {
      maxScore = 0
      matchScore = 0

      for each clue in secondary rule SR {
        maxScore += clue.weight //max possible score
        clueSim = clue.Similarity( p, candidate ) //clue match score
        if( clueSim >= clue.threshold ) //clue-level threshold
          matchScore += clue.weight //clue weight
      }
      matchScore = matchScore / maxScore //normalize
      classification = (matchScore >= SR.threshold) ? match : no match
      if(classification == match)
        add candidate to matchList
      else
        add candidate to noMatchList
    }
  }
}
```

Figure 4.7. This figure details how traditional non-fused deduplication uses clue-level
match-scores (and the clue-level thresholds) to specify a classification for records
contained in the primary candidate set.
Figure 4.8. This figure shows how clue-level fusion leverages the clues within a secondary rule. The SVM determines the classification of the current clue match-scores.

rules calculate and output the match score of every clue, even if that match score does not meet the original clue-level threshold.

First, for each deduplication rule, we create single clue match score examples by (1) calculating the similarity of the current probe and candidate gallery records for each of the individual clues in the current rule (using each clue’s similarity function), and (2) by outputting the individual clue-level match score, i.e., a clue label indicating which clue was used to calculate the match score, and whether or not this probe-gallery pair is an actual match, based on the FEBRL record ID of the probe and gallery records. We scale these single clue match scores to be in the range [0, 1] by dividing by the maximum similarity score for the current clue, where a value of 1 represents an exact match [16, 17]. Recall that when training an SVM, it is important to scale all of the input features so
that they have a common range, in order to keep any one input feature from dominating all other input features, simply due to different scales.

Next, for each deduplication rule, we concatenate the *single clue* match score examples of the same class, i.e., *match/no match*, to create *multi-clue*, i.e., entire rule, match score examples. We take the individual clue match scores with the corresponding clue label and *match/no match* label and concatenate the individual clue scores, as shown in Figure 4.9, wherein the specific content of each multi-clue example is determined by the set of clues that are used by the current deduplication rule.

Finally, we label the multi-clue training examples based on the classification label of the individual clue-level match scores. When creating training examples, we concatenate match scores of the same class, i.e., *match/no match*, so if all of the individual clue scores represent a match, the combined example also represents a match.

When training the best-fit SVMs for clue-level fusion, we train each SVM only on examples with no missing data. This means that if an SVM is trained to fuse clues $C_1$ and $C_2$, we train that SVM on examples that have match scores for both $C_1$ and $C_2$, i.e., the field data for clues $C_1$ and $C_2$ exist in both the probe and gallery record. We could train the clue-level fusion SVM on examples with missing data, but because we select the best-fit SVM based on the existence of clue data, the SVM will typically only be applied to examples where all of its feature data exists. See Figure 4.2 for an example of selecting

![Figure 4.9. A general example of the format of the training data for clue-level fusion.](image-url)
the best-fit SVM based on the existence of clue data. We choose to ignore training
examples with missing data because similar examples are unlikely to occur in the test
examples. Note that this differs from the training for rule-level fusion described in
Chapter 3, wherein we allow missing data in the training examples. When performing
rule-level fusion, the fusion occurs at the rule-level instead of at the clue-level, and the
overall rules are unaware of which individual clues have missing data. Therefore,
examples with missing data may occur in the rule-level fusion testing examples.

Selecting the Training Examples

After creating all of the multi-clue training examples for the current set of
deduplication rules and FEBRL-generated data set, we select a subset of examples on
which to train each clue-level fusion SVM. We train each clue-level fusion SVM
independently of any other SVMs in the system. We separately select a set of training
examples for each SVM and then train each SVM to solve the corresponding quadratic
programming optimization problem. No SVM is affected by any other SVM in the
system. As mentioned in Chapter 3, we do not want to over-fit the SVM to the training
data. Instead, we want the SVM to generalize properly so that it can correctly classify
unseen (testing) examples. In an attempt to avoid over-fitting the SVM, we select the
training examples randomly with replacement. We do not explicitly select only unique
training examples, and therefore it is possible that the same training example is selected
more than once while training the same SVM [16, 17].

We select up to a user-specified number of training examples. In our experiments,
we typically used anywhere between 100 and 500 training examples. To account for the
variability in example selection, we perform ten runs of training and keep the trained
SVM with the highest cross-validation accuracy over all training runs. We describe this cross-validation accuracy calculation in the next section. See the Empirical Results section for a discussion of the specific number of training examples used in each experiment.

**Selecting the SVM Parameters**

In our experiments, we implemented each best-fit fusion SVM via LIBSVM [16, 17]. Specifically, each fusion SVM employs a radial basis function (RBF) kernel $e^{-\gamma|v-u|^2}$. Just as when training the rule-level fusion SVM as described in Chapter 3, when training each clue-level fusion SVM, we choose the appropriate $\gamma$-value and constraints-violation cost $C$ at run-time by performing $k$-fold (stratified) cross-validation on the current set of training examples. We perform a grid search to find these parameter values, letting $\gamma$ be in the range $[2^{-15}, 2^3]$ with a step size of 4, and letting $C$ be in the range $[1, 2^{10}]$ with a step size of 4 [16, 17]. We keep the parameter values for $\gamma$ and $C$ that yield the highest five-fold cross-validation accuracy on the selected set of training examples. See Section *Selecting the SVM Parameters* in Chapter 3 for a description of this grid search and using cross-validation accuracy to determine the appropriate SVM parameter values. After determining the best SVM parameter values using the grid search described in Chapter 3, we re-train the SVM on the entire set of selected training examples, i.e., all five folds of training examples [16, 17].
Clue-Level SVM Fusion

We train each SVM to fuse the single-clue match scores and output one overall classification decision for the current rule:

\[ CL - SVM_{SR}: C_1 \times C_2 \times C_3 \times C_4 \ldots \times C_n \rightarrow \{match/no\ match\} \]

where \( C_1 \) is the real-valued match score for clue 1 and the current secondary rule \( SR \) contains some number of clues (represented by \( n \)). As described in the previous sections, we train one SVM for each unique subset of the clues in the current rule. \( CL - SVM_{SR} \), shown here, represents one unique subset of clues in the current secondary rule, \( SR \).

Experiment

Data Sets

For our clue-level fused deduplication experiments, we used FEBRL [20, 21] to generate several data sets that conform to the GJXDM schema [22]. See Figure 2.5 for an example of the GJXDM schema. As mentioned previously, FEBRL (1) generates an original set of records, based on real-world Australian census data, and next (2) generates duplicate records by adding noise to the original records. FEBRL modifies the original records, by introducing real-world field modifications, i.e., errors or noise, including typographical errors, transposed letters and missing data [20, 21]. For a more thorough description of how FEBRL generates the original and duplicate records, see Chapter 2 and Chapter 3.

To demonstrate the efficacy of our clue-level fused deduplication technique, we specifically generated FEBRL data sets that contain missing data, i.e., partial records. As
shown in Table 4.1, we generated data sets ranging from 0% missing data to 45% missing data. The *Percentage Missing* column shown in Table 4.1 represents the average percentage of all missing fields in the data set regardless of which specific fields are missing. We did not want to bias the results of our experiment by limiting the missing data to only those fields that we do not care about, e.g., non-clue fields.

Further, the specified percentage missing is averaged over all records in the data set, including both probe and gallery records. For example, the data set described in the third row of Table 4.1, with 30% missing data, contains probe records with 30% missing field data and gallery records with 30% missing field data, on average. We did not restrict the missing data to specific fields, so it is possible that when comparing two records, they will be missing different fields, thus increasing the difficulty of identifying duplicate records.

<table>
<thead>
<tr>
<th>Modifications per Record</th>
<th>Modifications per Field</th>
<th>Data Set Similarity</th>
<th>Percentage Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>82.60%</td>
<td>0%</td>
</tr>
<tr>
<td>20</td>
<td>9</td>
<td>66.14%</td>
<td>15%</td>
</tr>
<tr>
<td>21</td>
<td>10</td>
<td>54.38%</td>
<td>30%</td>
</tr>
<tr>
<td>28</td>
<td>11</td>
<td>46.52%</td>
<td>45%</td>
</tr>
</tbody>
</table>

Table 4.1. FEBRL-Generated Data Sets.

Just as in Chapter 3, we calculate the *average similarity* of each FEBRL-generated data set by calculating the average edit distance between each original FEBRL record and its corresponding duplicate records. For a more thorough description of this data set similarity calculation, see Chapter 3.

Each data set described in Table 4.1 contains 1000 probe records and 1500 gallery
records. Each row in Table 4.1 represents five data sets generated with the same FEBRL parameters, i.e., the number of modifications per field and per record. We averaged the Data Set Similarity and Percentage Missing over these five data sets. The average values shown in Table 4.1 are consistent over all five runs. For each data set, the standard deviation is less than 0.3%.

**Method**

After creating the initial set of primary and secondary rules and specifying the rule- and clue-level thresholds, we purposefully did not tune them, allowing us to demonstrate that our multi-SVM technique is not dependent on any manual tuning [1, 2, 3]. When performing clue-level best-fit SVM fusion, a $CL - MM$ executes each primary rule $PR$ in the $PRS$ to generate a candidate set $CS$. After the primary rules generate a $CS$, the $SR$s execute their clues to produce clue-level match scores for each candidate, and the $CL - MM$ uses those match scores as inputs to determine which best-fit SVM to use. Then, the best-fit SVM fuses those match scores to determine one overall classification for the record in question for the current rule. (See Figure 4.8.) If the hypothesis holds that this approach reduces the need for manual tuning, performing any kind of tuning of the rule- or clue-level thresholds via manual means would reduce the efficacy of the experiment.

As with the rule-level $RL - MM$ technique presented in Chapter 3, this $CL - MM$ technique also leverages primary and secondary rules. A brief discussion of the usage of these rules, within the scope of the $CL - MM$ technique follows.
Primary Rules

First, the PRs are executed, to generate a candidate set (CS). The $CL-MM$ then passes the CS to the secondary rules (SR). To allow for a direct comparison between the accuracy of the $RL-MM$ presented in Chapter 3 and the $CL-MM$ outlined in this chapter, the $CL-MM$ (shown here) uses the same set of primary and secondary rules as the $RL-MM$ experiments in Chapter 3. Figures 4.10-4.13 summarize these primary rules.

The $CL-MM$ uses the primary rules, shown in Figures 4.10-4.13, to generate a list of possible matching candidate records: in other words, the candidate set (CS).

```c
PrimaryRule1 (p) { // is a probe record
    create empty candidateList

    for each g in the gallery {
        if (p.passportID == g.passportIDfirst 3 chars)
            if (p.passportCountry == g.passportCountryfirst 3 chars)
                if (p.visaID == g.visaIDfirst 3 chars)
                    if (p.visaType == p.visaTypefirst 3 chars)
                        add g to candidateList
    }

    Return candidateList
}
```

Figure 4.10. Primary Rule 1. The primary rules generate the primary candidate set.
PrimaryRule2 (p) \{  //p is a probe record
    create empty candidateList

    for each g in the gallery {
        if (p.passportID == g.passportID\text{first 3 chars})
            if (p.gender == g.gender)
                if (p.nationality == g.nationality\text{first 3 chars})
                    if (p.dateOfBirth exists)
                        if (g.dateOfBirth exists)
                            add g to candidateList
    }

    Return candidateList
\}

Figure 4.11. Primary Rule 2.

PrimaryRule3 (p) \{  //p is a probe record
    create empty candidateList

    for each g in the gallery {
        if (p.gender == g.gender)
            if (p.nationality == g.nationality\text{first 3 chars})
                if (p.surname == g.surname\text{first char})
                    if (p.givenName == g.givenName\text{first char})
                        if (p.orgName == g.orgName\text{first char})
                            add g to candidateList
    }

    Return candidateList
\}

Figure 4.12. Primary Rule 3.
Secondary Rules

Once candidate set $CS$ is prepared, the $CL-MM$ executes the clues contained in the first $SR$, generating a match score $MS$ for each clue. If the clues for the first $SR$ can generate a useful set of match scores with cardinality of at least two, the $CL-MM$ selects the SVM leveraging the highest number of clue match scores, i.e., clues for which data exists, and classifies the current probe-candidate pair as a match/no match. As with the primary rules, the same set of secondary rules used in the $RL-MM$ experiment described in Chapter 3 are used here. Figures 4.14-4.16 summarize these secondary rules. Just as in the rule-level fusion experiments presented in Chapter 3, we did not manually tune these rules or their thresholds, to demonstrate that our fused deduplication technique reduces the need for manual tuning.
Figure 4.14. Secondary Rule 1. The secondary rules calculate the record match-scores.

```plaintext
SecondaryRule1 (p, c) { // p is a probe, c is a candidate match
    matchScore = 0

    if (EditDistance(p.passportID, c.passportID) ≤ 1)
        matchScore += 10 // clue weight
    if (EditDistance(p.passportCountry, c.passportCountry) ≤ 1)
        matchScore += 5
    if (EditDistance(p.visaID, c.visaID) ≤ 1)
        matchScore += 8
    if (EditDistance(p.visaType, c.visaType) ≤ 1)
        matchScore += 2

    if (matchScore ≥ 20) // rule-level threshold
        return (matchScore/25)
    else
        return 0
}
```

Figure 4.15. Secondary Rule 2.
Figure 4.16. Secondary Rule 3. This rule contains ambiguous fields, such as proper names.

Clue Match-Score Level Fusion

Our clue-level fused deduplication technique operates in much the same manner as the non-fused rule-based deduplication system described above. However, our fused deduplication technique does not rely on any manually tuned clue- and rule-level thresholds. Instead, we employ learning-based fusion [1, 2] to the clues within each secondary rule, as described in Figure 4.8. Unlike traditional non-fused deduplication, we do not apply the secondary clue- and rule-level thresholds when determining a match using clue-level fusion; instead, we use the trained SVM to determine if the current clue scores represent a match.

We trained the clue-level fusion SVMs for the set of specific deduplication rules shown in Figures 4.10-4.16 in the same manner as described in the SVM Training section,
above. In our experiments, we trained each SVM on 200 training examples selected from a set of well over 1.5 million training examples, created by comparing 1000 probe records to 1500 gallery records, one clue at a time. Empirically, using 200 training examples proved sufficient for training the clue-level fusion SVMs; increasing the number of training examples did not improve the accuracy of the clue-level fusion SVMs.

**Testing Data**

We used FEBRL to generate multiple data sets containing 1500 original and 1000 duplicate records [20, 21], described in Table 4.1. We trained the SVM on one of these FEBRL-generated data sets. To test the clue-level fusion SVMs as part of the overall fused deduplication system, we generated additional FEBRL data sets, using the same general parameters as the FEBRL-generated data set used to train the clue-level fusion SVMs. Just as with our rule-level fusion experiments described in Chapter 3, we specifically trained on one data set and tested on a different data set to avoid biasing the experimental results with the training data.

As discussed in Chapter 3, because FEBRL uses specific real-world probabilities to randomly select the field to modify and the type of modification to perform, no two FEBRL-generated data sets will be exactly the same. However, multiple FEBRL data sets that are generated with the same real-world probabilities and the same parameters, including the number and distribution of duplicate records, will have similar types and amounts of noise [20, 21]. Because we are performing fusion on the match scores generated by the individual clues rather than on the specific contents of the record fields, we can train the clue-level fused deduplication system on one FEBRL-generated data set and test the system on another FEBRL-generated data set [1, 2].
Testing the Clue-Level Fused Deduplication

When testing the clue-level fusion SVMs, we proceed much as in the training phase: we pass the current data set through the deduplication rules shown in Figures 4.10-4.16, and determine if any of the records are duplicates. As shown in Figure 4.8, the clue-level fusion SVMs operate within the deduplication system. We simply use the classification decision of the selected clue-level fusion SVM to determine whether the current probe-gallery record pair represents an actual match for the current secondary rule. The rest of the deduplication system, i.e., the primary and secondary rules and how they are applied to the probe and gallery records, operate in the same manner as in the original non-fused deduplication.

The main difference between the training and testing phases is that when testing, we do not already know the label for the current vector of clue match scores. Instead, we use the selected best-fit SVM to predict the classification label of match/no match for the current probe-candidate pair:

\[ CL - SVM_{SR}: C_1 \times C_2 \times C_3 \rightarrow \{match/no match\} \]

As described in Figure 4.8, for the current secondary rule, we generate the single clue match scores by separately applying each clue in the current secondary rule to the current probe-candidate pair. See Figures 4.14-4.16 for a pseudo-code description of the secondary rules. We scale the single clue match scores to be in the range [0, 1] by dividing by the maximum similarity score for the current clue [16, 17]. Next, we concatenate the single clue match scores to create a multi-clue, i.e., entire rule, score
vector as shown in Figure 4.17.

Finally, we perform fused multi-clue deduplication, using an appropriately trained SVM: we apply the selected best-fit clue-level fusion SVM to the current multi-clue score vector, to determine its class label, i.e., match/no match. The clue-level SVM outputs one overall classification decision, indicating match/no match for the current probe-candidate pair. Unlike traditional deduplication, we do not apply the secondary rule-level thresholds when determining a match using clue-level fusion; instead, we use the trained SVM to determine if the current scores represent a match.

![Figure 4.17. This figure represents the format of the data inputted into the best-fit SVM. The SVM predicts the classification (match/no match) of the probe-candidate pair.](image)

**Empirical Results**

Figures 4.18-4.21 present the experimental results generated by the $CL - MM$ on the FEBRL-generated test data sets presented in Table 4.1. In comparison to the $RL - MM$ in Chapter 3, the $CL - MM$ experiments presented here highlight the percentage of missing data fields. To directly compare the $RL - MM$ presented in Chapter 3 and the $CL - MM$ presented in this chapter, we applied each of these fused deduplication techniques to the data sets summarized in Table 4.1.

The clue-level fused deduplication results shown in Figures 4.18-4.22 were achieved with 200 training examples. The rule-level fused deduplication results shown in Figures 4.18-4.22 were achieved with 300 training examples, the same number of
training examples used in Chapter 3. All deduplication results shown in Figures 4.18-4.22 are averaged over five testing runs per FEBRL-generated data set presented in Table 4.1. We performed five runs of testing, using similar FEBRL-generated data created with the same FEBRL parameters, wherein each of these data sets has the specified average data similarity, as described in Table 4.1. Each data set contains 1000 probe records and 1500 gallery records. When performing deduplication, the system compares each probe record against each candidate gallery record. This means that for each data set, the deduplication systems performs up to 1.5 million total comparisons for all probes, i.e., 1000 probe records compared against 1500 gallery records.

As described in Table 4.1, we applied our fused deduplication technique to data sets with missing field data: from 0% missing data to 45% missing data. The missing field data increased as the experiments proceeded, hence the drastic sensitivity and specificity values recorded for the first data set with no missing data (Figure 4.18) and the final data set containing 45% missing field data (Figure 4.21). The percentage of missing data indicates the percentage of fields that are empty, averaged over all records in the data set (including both probe and gallery records).

![Table 4.1](image)

**Figure 4.18.** Experimental results with no missing data. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).
Figure 4.19. Experimental results with 15% missing data. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).

Figure 4.20. Experimental results with 30% missing data. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).

Figure 4.21. Experimental results with 45% missing data. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).
0% Missing Data

In the first experiment (see Figure 4.18), even though the traditional non-fused technique performs mildly well at 0.5267 sensitivity, the two fusion techniques both perform quite well. In fact, even with a high noise level generated by performing up to ten modifications per record, and up to five modifications per data field, for an overall similarity of 82.60% (see Table 4.1), Figure 4.18 shows that the rule-level fusion technique still generated a sensitivity of 0.9869. This shows that when no data is missing, the rule-level fusion technique performs extremely well and is able to overcome the need for manual rule tuning.

On average, the $CL-MM$ did not identify any false negatives on the data sets used for this experiment, thus resulting in a sensitivity of 1, i.e., the $CL-MM$ correctly identified all actual duplicates. This indicates that our clue-level fusion technique also reduces the need for manual rule tuning. However, its sensitivity value of 1 is barely above that generated by the $RL-MM$ presented in Chapter 3. This speaks to the efficacy of our previous technique when the amount of missing data is minimal or non-existent in the data set under consideration. Unfortunately, as previously presented, real-world data does suffer from missing data [6, 7, 10, 20, 21]. Figures 4.19-4.21 represent the values attained when the test data contains missing data at 15%, then 30%, and finally 45%, along with an increasing amount of additional noise.

15% Missing Data

When 15% of the data is missing along with only a 66.14% data set similarity, the sensitivity of the traditional non-fused technique drops from the 0.5267 reported in Figure 4.18, to the 0.2681 value presented in Figure 4.19. This indicates that as the
amount of missing data increases, the traditional non-fused deduplication system fails to identify as many duplicate records. The rule-level fusion technique is also not immune to the missing data, as its sensitivity dropped to 0.8211 with 15% missing data. The clue-level fusion presented here in Chapter 4, however, continues to perform well, with a sensitivity value of 0.9975, and an average false negative value of only 9, whereas rule-level fusion averaged 643.2 false negatives for this data set, and traditional non-fused averaged a false negative value of 2631.8. Even at only 15% missing data, the efficacy of the clue-level best-fit SVM presented in this chapter stands out. Recall that our SVMs are trained for sensitivity, because of the cross-validation accuracy test that we use. Therefore, we expect the sensitivity of the fused deduplication techniques to be greater than that of the traditional non-fused deduplication, as shown in Figure 4.19.

30% Missing Data

Figure 4.20 shows the deduplication results with 30% of the data missing. As the percentage of missing data doubles from 15% to 30%, the number of true positives identified by the traditional technique drops by over 30%. Even though the average specificity of the traditional deduplication remains high at 0.9927, its sensitivity drops to just 0.1836. Again, this indicates that as the amount of missing data increases, the traditional non-fused deduplication system fails to identify as many actual duplicates.

The higher level of missing data also affects the rule-level fusion results. Even though higher than the traditional technique's average sensitivity, the rule-level fusion's sensitivity only attains a value of 0.6536 with 30% missing data. The $CL - MM$ technique continues to perform well, even at 30% missing data, with an average sensitivity of 0.9672.
As shown in Figures 4.18-4.21, as the amount of missing data increases, the
difference in the deduplication results produced by the different techniques becomes
more significant. The traditional and rule-level fusion techniques produce lower
sensitivity than the clue-level fusion technique when faced with missing data. This
indicates that clue-level fusion is better able to overcome the missing field data and
identify actual duplicate records.

45% Missing Data

With 45% of the data missing and the data similarity of the remaining data at only
46.52%, the disparity between the three techniques’ sensitivity becomes even more
pronounced. Traditional deduplication achieves a sensitivity value of 0.1903, while rule-
level fusion attains 0.6409. On the other hand, clue-level best-fit SVM fusion continues
to perform well with a sensitivity value of 0.8252 (see Figure 4.20). Even when almost
half of the data is missing and half of the remaining data contains noise in its data fields
up to and including 28 modifications per record, with up to 11 modifications per field (as
described in Table 4.1), the $CL-MM$ continues to perform at a high level. On this
difficult data set, clue-level fusion not only identifies more true positive duplicate
records, but also identifies fewer false negatives than either traditional or rule-level
fusion deduplication. This indicates that the clue-level fused deduplication better adapts
to missing data. Because it fuses around the missing field data whenever possible, best-
fit clue-level fusion is better able to overcome the impediment of missing data than either
rule-level fusion presented in Chapter 3 or the traditional non-fused deduplication.

Figure 4.22 provides an aggregate view of the results presented in Figures 4.18-
4.21. These results show the performance of each technique, not only in relationship to
Figure 4.22. The different techniques applied to each data set shown in Table 4.1, with the x-axis emphasizing the percentage of missing data in each experimental data set.

Each other, but this figure purposefully highlights the amount of missing data in each data set, starting with 0% and culminating in the final data set with 45% missing data.

Figure 4.22 clearly shows that as the amount of missing data increases, the efficacy of all three techniques degrades. In the final experiment, with 45% of the data missing, clue-level fusion still performs at a sensitivity level of better than 82%. Conversely, rule-level fusion performs at only 64.09% sensitivity while the traditional non-fused technique performs at only 19.03% sensitivity.

Note that the sensitivity of the traditional non-fused deduplication slightly improved from 18.36% to 19.03% when the missing data increased from 30% to 45%, as shown in Figure 4.22. When using FEBRL to generate the data sets, to avoid biasing the results, we do not specifically control which fields contain missing data or other noise.

The difference in the sensitivity of the traditional deduplication technique on the data set with 30% missing data (Figure 4.20) versus the data set with 45% missing data (Figure
4.21) is likely due to where the missing data actually occurred in each data set. For example, for the data set with 45% missing data, the clues used by the original deduplication rules may not be missing as much data as for the data set with 30% missing data.

Overall, as the data-quality degraded, the performance differences among clue-level fusion, rule-level fusion, and traditional deduplication became clearer. Even though clue-level fusion technique's performance degraded as the amount of missing data increased, it continued to outperform both the rule-level fused deduplication and the traditional non-fused deduplication. This higher level of performance is due to the intrinsic nature of the $CL-MM$. This technique accounts for missing data in any of the fields identified by any of the clues. Whether the missing data is found in the gallery record, or in the probe record, the performance of the $CL-MM$ is the same: the $CL-MM$ overcomes missing data in any of the records. Using best-fit multi-SVM fusion at the clue-level allows the $CL-MM$ to fuse around the missing data, resulting in a higher sensitivity value than either traditional deduplication or the $RL-MM$ technique presented in Chapter 3, can produce as the amount of missing data increases. As previously discussed, real-world data sets do contain missing data, and the $CL-MM$ technique presented in this chapter proves effective in identifying duplicate records when missing data is prevalent.

Discussion of Strengths and Weaknesses

The empirical results show that when performing rule-based deduplication specifically in the presence of missing data, the clue-level best-fit fusion technique presented in this chapter performs much more robustly than either traditional non-fused
deduplication or even the rule-level fusion technique presented in Chapter 3.

It is encouraging that rule-level fusion still performs somewhat efficiently when data is purposefully missing, lending credence to the hypothesis that SVM fusion is an effective tool to replace the manual tuning of domain expert derived rules, but more encouraging still are the results generated by clue-level best-fit SVM fusion.

The rule-level fusion is more simplistic, and proved easier to implement. It did not require nearly as much training, as there was only one SVM to train, whereas with the multi-SVM approach presented here, the power set of all of the unique clues used in the secondary rules determines the number of SVMs to train. Potentially, if the set of matching rules is large and if the feature sets considered by the secondary rules have little to no overlap between them, the number of best-fit SVMs could be difficult to manage, or more likely, could take a large amount of time to correctly train. However, the time/improvement tradeoff is negligible in comparison to the time that might be required to manually tune the clue- and rule-threshold levels for each secondary rule, as the SVM training is fairly automated. All that needs to be selected is the number of training examples to use.

The initial experimental results presented in Figure 4.18 show that when there is no missing field data within the data set, clue-level fusion does not perform markedly better than rule-level fusion, bringing into the question the viability of such a solution due to the higher implementation cost. However, the results presented in Figures 4.19-4.21 clearly demonstrate the efficacy of clue-level best-fit multi-SVM fusion in the presence of missing data. As more and more of the data disappeared, clue-level fusion continued to perform robustly, whereas our rule-level fusion technique started to suffer. Even
though more complex to implement, when executed on data sets representing the real-world concern of missing data, the $CL-MM$ technique presented here in Chapter 4 outperformed the $RL-MM$ technique presented in Chapter 3.

Regardless of the chosen fusion technique, or if traditional rule-based deduplication is performed, the efficacy of the original deduplication rules is still an issue. Ineffective primary and secondary rules will limit the gains provided by using a SVM for classification. In the following chapter, we apply our clue- and rule-level fusion techniques to additional sets of deduplication rules.
CHAPTER 5

ADDITIONAL EXPERIMENTS

Overview

The rule-level fusion technique \((RL-MM)\) presented in Chapter 3 shows empirically that applying rule-level fusion to data deduplication yields distinctly higher average accuracy than the comparable traditional non-fused rule-based deduplication system (see Figure 3.26). In fact, as the similarity between the data records decreased, the performance difference between the two techniques became quite distinct. The \(RL-MM\) achieved this marked improvement in deduplication accuracy without any manual tuning of the clue- and rule-level thresholds. Instead, the \(RL-MM\) employed a single SVM to fuse the individual rule-level match-scores, generated by the secondary deduplication rules, into one overall classification indicating whether the current probe-candidate pair matched. This single SVM replaced the need to employ a high level of manual effort to iteratively tune the deduplication rules, thus making the deployment of a rule-based system much more practical for a real-world setting [3, 7, 20, 21].

The clue-level best-fit multi-SVM fusion technique \((CL-MM)\) presented in Chapter 4 is specifically designed to perform data deduplication on records with missing data, by using a best-fit SVM to fuse around missing clue data. When compared to the rule-level fusion employed in Chapter 3, this approach is more robust and complete, as the results presented in Chapter 4 show (see Figure 4.22). The experimental results in Chapter 4 report that rule-level fusion did not perform as well as the best-fit multi-SVM approach when the data set contains missing data. In fact, as the missing data increased
from 0% to 45%, the $CL - MM$ continued to perform well, whereas the performance of the $RL - MM$ technique on the same set of rules degraded much more rapidly and failed to achieve the same level of sensitivity.

The results reported in Chapter 4 demonstrated that the clue-level multi-SVM approach provides more accurate deduplication results than rule-level fusion when dealing with a data set containing a significant amount of missing data, as well as continuing to reduce the need to manually tune the clue- and rule-level thresholds. The multi-SVM approach is arguably more complex to implement than the rule-level fusion, as discussed in Chapter 4, but the robust nature of the technique outweighs many negative aspects of the implementation.

Experiments

To lend further credence to our fused deduplication techniques, we ran additional experiments for both rule-level fused deduplication ($RL - MM$) and best-fit multi-SVM clue-level fused deduplication ($CL - MM$). We did this by using two new sets of human intuited primary and secondary rules.

Data Sets

In order to better compare these two new rule sets to the set of rules presented in Chapter 3 and Chapter 4, we trained and tested these two new rule sets on two of the data sets used in our previous experiments. We chose to use one data set from Chapter 3 with 72% similarity and one data set from Chapter 4 with 54.38% similarity. These data sets proved challenging, containing significant amounts of noise and missing field data. Table 5.1 shows the chosen data sets used for these additional experiments. Tables 3.1
and 4.1 show these data sets presented here in relation to the original data sets chosen for our rule- and clue-level fusion experiments, respectively. Each data set described in Table 5.1 contains 1000 probe records and 1500 gallery records. Recall that each row in Table 5.1 represents five data sets generated with the same FEBRL parameters, i.e., the number of modifications per field and per record [20, 21]. Just as in the previous chapters, we averaged the Data Set Similarity and Percentage Missing over these five data sets. The average values shown in Table 5.1 are consistent over all five runs. For each data set, the standard deviation is less than 0.5%.

When compared with the other original data sets from Chapters 3 and 4, neither data set shown in Table 5.1 contains the highest level of similar data, nor do they contain the lowest level of missing data. Instead, they both introduce a challenging level of noise into the data fields, including extra whitespace, missing and transposed characters, etc., along with a not insignificant percentage of missing data [20, 21].

Table 5.1. The Data Sets Used for the Additional Experiments.

<table>
<thead>
<tr>
<th>Modifications per Record</th>
<th>Modifications per Field</th>
<th>Data Set Similarity</th>
<th>Percentage Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>72.00%</td>
<td>14%</td>
</tr>
<tr>
<td>21</td>
<td>10</td>
<td>54.38%</td>
<td>30%</td>
</tr>
</tbody>
</table>

Two Additional Sets of Primary and Secondary Rules

Acting as a domain expert [3], we defined and created two new sets of primary and secondary rules. We identified and included discriminatory fields, or sets of fields as appropriate, for both the primary and secondary rules.
Rule Set A

Purposefully meant to be different and to function in a more discriminatory fashion than the rule/clue sets used in Chapters 3 and 4, this set of rules still needed to successfully perform both the initial blocking and secondary matching functions. As an additional change from the original rule/clue sets, this rule set only contains two primary and two secondary rules, rather than the original count of four primary rules and three secondary rules. Additionally, the record fields identified for the creation of the candidate sets via the primary rules, as well as the requisite number of required matching characters, is different from those used in the set of primary rules presented in Chapters 3 and 4. Also, the allowable edit distance for each individual clue, as well as the clue weights and rule thresholds, are different from those used in the initial set of secondary rules. Figures 5.1-5.4 contain pseudo code for these new primary rules in Rule Set A.

```java
PrimaryRuleA1 (p) {
  Create empty candidateList
  For each g in the gallery {
    if(p.passportID == g.passportID first 5 chars) // up from 3
      if(p.nationality == g.nationality first 3 chars) // in place of passportCountry
        if (p.visaID == g.visaID first 4 chars) // up from 3
          if(p.surname == g.surname first 2 chars) // NEW
            add g to candidateList
  }
  Return candidateList
}
```

Figure 5.1. Primary Rule 1 in Set A.
PrimaryRuleA2 (p) {
    Create empty candidateList
    For each g in the gallery {
        if(p.passportID == g.passportID first 4 chars) // up from 3 (but not 5)
            if(p.visaType == g.visaType first 2 chars) // down from 3
                if(p.surname == g.surname first 2 chars) // NEW
                    if(p.gender == g.gender) // NEW
                        if(p.orgName == g.orgName first 3 chars) // up from 1
                            add g to candidateList
    }
    Return candidateList
}

Figure 5.2. Primary Rule 2 in Set A.

SecondaryRuleA1 (p, c) {
    matchScore = 0
    if (EditDistance (p.passportID, c.passportID) <= 1)
        matchScore += 10 // clue weight
    if (EditDistance (p.surname, c.surname) <= 1) // same on passportID
        matchScore += 9 // clue weight
    if (EditDistance (p.visaType, c.visaType) <= 1) // less weight here
        matchScore += 6 // clue weight
    if (EditDistance (p.nationality, c.nationality) <= 2) // less weight here
        matchScore += 4 // clue weight
    if (matchScore >= 29)
        return (matchScore/29)
    else
        return 0
}

Figure 5.3. Secondary Rule 1 in Set A.
Figure 5.4. Secondary Rule 2 in Set A.

```
SecondaryRuleA2 (p, c) {
    matchScore = 0

    if (EditDistance (p.visaID, c.visaID) <= 1)
        matchScore +=10 // clue weight
    if (EditDistance (p.passportID, c.passportID) <= 1)
        matchScore +=8 // clue weight
    if (EditDistance (p.visaType, c.visaType) <= 1) // same on passportID
        matchScore +=6 // clue weight
    if (p.gender == c.gender) // less weight here
        matchScore +=2 // clue weight

    if (matchScore >= 26)
        return (matchScore/26)
    else
        return 0
}
```

Rule Set B

Neither as intelligent nor as discriminatory as the original rules/clues used in Chapters 3 and 4, or in Rule Set A detailed above, this rule set provides a framework to produce experimental results to further support the concern that even though match-score fusion improves the accuracy of rule-based deduplication, if the chosen rule/clue combinations are poor, the efficacy of the fusion system will suffer accordingly. This rule set contains three primary rules and three secondary rules. Figures 5.5-5.10 contain the pseudo code for these less-discriminatory rules.
PrimaryRuleB1 (p) {
    Create empty candidateList

    For each g in the gallery {
        if(p.passportID == g.passportID first 2 chars)
            if(p.nationality == g.nationality first 1 chars)
                add g to candidateList
    }
    Return candidateList
}

Figure 5.5. Primary Rule 1 in Set B.

PrimaryRuleB2 (p) {
    Create empty candidateList

    For each g in the gallery {
        if(p.passportID == g.passportID first 1 chars)
            if(p.surname == g.surname first 2 chars)
                add g to candidateList
    }
    Return candidateList
}

Figure 5.6. Primary Rule 2 in Set B.
PrimaryRuleB3 (p) {
    Create empty candidateList

    For each g in the gallery {
        if(p. surname == g. surname first 1 chars)
            if(p.nationality == g.nationality first 1 chars)
                if (p.gender == g.gender)
                    add g to candidateList
    }
    Return candidateList
}

Figure 5.7. Primary Rule 3 in Set B.

SecondaryRuleB1 (p, c) {
    matchScore = 0

    if (EditDistance (p.visaID, c.visaID) <= 1)
        matchScore +=10  // clue weight
    if (EditDistance (p.surname, c.surname) <= 1)
        matchScore +=6  // clue weight
    if (EditDistance (p.givenName, c.givenName) <= 1) // same on passportID
        matchScore +=4  // clue weight
    if (p.gender == c.gender) // less weight here
        matchScore +=2  // clue weight

    if (matchScore >= 22)
        return (matchScore/22)
    else
        return 0
}

Figure 5.8. Secondary Rule 1 in Set B.
SecondaryRuleB2 (p, c) {
    matchScore = 0
    if (EditDistance (p.passportCountry, c.passportCountry) <= 1)
        matchScore +=10 // clue weight
    if (EditDistance (p.surname, c.surname) <= 1) // less weight here
        matchScore +=8 // clue weight
    if (EditDistance (p.nationality, c.nationality) <= 2) // less weight here
        matchScore +=7 // clue weight
    if (EditDistance (p.dateOfBirth, c.dateOfBirth) <= 2) // less weight here
        matchScore +=5 // clue weight
    if (matchScore >= 30)
        return (matchScore/30)
    else
        return 0
}

Figure 5.9. Secondary Rule 2 in Set B.

SecondaryRuleB3 (p, c) {
    matchScore = 0
    if (EditDistance (p.nationality, c.nationality) <= 2) // less weight here
        matchScore +=7 // clue weight
    if (EditDistance (p.givenName, c.givenName) <= 1) // same on passportID
        matchScore +=4 // clue weight
    if (EditDistance (p.surname, c.surname) <= 1)
        matchScore +=6 // clue weight
    if (EditDistance (p.orgName, c.orgName) <= 1)
        matchScore +=2 // clue weight
    if (matchScore >= 19)
        return (matchScore/19)
    else
        return 0
}

Figure 5.10. Secondary Rule 3 in Set B.
Training SVMs for the New Rules

As detailed in Chapters 3 and 4, we train each SVM on the deduplication examples generated by applying the current set of deduplication rules to a FEBRL-generated data set. Thus, the SVM training examples are dependent upon the current set of deduplication rules. When applying our fused deduplication technique to a new set of deduplication rules, we train the fusion SVM anew.

When trained appropriately, the SVM generalizes properly, and therefore no retraining is necessary for every new data set, within reason. If the data set has changed drastically, such that the data set used to train the SVM no longer resembles the testing data set at all, the SVM should be re-trained. In order for the SVM to generalize properly, we avoid over-fitting the SVM to the training data by sampling the training examples in a sufficiently random manner, as described in Chapters 3 and 4. Just as in Chapters 3 and 4, we performed cross-validation on the selected training examples, and kept the SVM parameters $\gamma$ and $C$ that produced the highest cross-validation accuracy.

We trained the new SVMs used in this chapter on 300 training examples for rule-level fusion and 200 training examples for clue-level fusion. To account for the variability in the random example selection, we performed ten runs of training and kept the SVM that achieved the highest cross-validation accuracy over all runs for the current rule set and data set.

It is important to note that even though we used the same probability distributions when creating the data sets for these additional experiments as we did for the corresponding data sets used in the previous chapters, we did not explicitly instruct FEBRL to affect one field more than any other field. Nor did we explicitly tell FEBRL
which type of noise to introduce other than the target missing data percentages. Using FEBRL to generate the data sets based on real-world probabilities helps us to avoid introducing artificial and problematic biases [20, 21].

Empirical Results

Rule Set A

Consider the results obtained from running Rule Set A on the FEBRL-generated data sets shown in Table 5.1, presented in Figures 5.11-5.13. Figures 5.11 and 5.12 show not only the results recorded by running our rule-level and clue-level fusion experiments, but they also present the results obtained by, once again, executing the rule set in the traditional non-fused technique to provide a better means to compare and contrast the experimental results.

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Rule-Level Fusion</th>
<th>Clue-Level Fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True Positive: 308.4</td>
<td>False Positive: 6947</td>
<td>True Positive: 2224.2</td>
</tr>
<tr>
<td></td>
<td>False Negative: 3287.6</td>
<td>True Negative: 1488457</td>
<td>False Negative: 1371.8</td>
</tr>
<tr>
<td></td>
<td>Sensitivity: 0.0850</td>
<td>Specificity: 0.9954</td>
<td>Sensitivity: 0.6185</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Figure 5.11. The results obtained by Rule Set A with 30% missing data and a similarity of 54.38%. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).
Figure 5.12. The results obtained by Rule Set A with 14% missing data and a similarity of 72%. This data set contains a total of 3644 duplicates and 1495356 non-duplicates (for all 1000 probe and 1500 gallery records).

<table>
<thead>
<tr>
<th>Rule Level Fusion</th>
<th>Traditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive: 2196.8</td>
<td>True Positive: 487</td>
</tr>
<tr>
<td>False Positive: 61767</td>
<td>False Positive: 0</td>
</tr>
<tr>
<td>False Negative: 1447.2</td>
<td>False Negative: 3157</td>
</tr>
<tr>
<td>True Negative: 1433589</td>
<td>True Negative: 1495356</td>
</tr>
</tbody>
</table>

Sensitivity 0.6030, Specificity 0.9587, Sensitivity 0.9177, Specificity 0.8956

Figure 5.13. This figure shows Rule Set A applied to both data sets: one data set containing 30% missing data (with 52.38% similarity), and a second data set with 14.03% missing data (with 72% similarity).

Rule Set A: 52.38% Similarity, 30% Missing

When data quality is poor, all three deduplication techniques suffer. Figure 5.11 demonstrates that with low data similarity and a high percentage of missing data, all three techniques suffer, when leveraging Rule Set A. When determining the efficacy of either rule- or clue-level fusion, it is important to note how badly traditional non-fused deduplication performs in comparison to our two fusion techniques. The rules and clue
combinations in Rule Set A contain higher thresholds and stricter clue functions. There are fewer leniencies allowed when compared with the original rule set used in Chapters 3 and 4. This attempt at introducing a stricter rule set negatively affects the number of duplicate records identified with the initial rules. When applied via traditional non-fused deduplication, Rule Set A only identified an average of 308.4 of the duplicate records, for an accuracy of 8.58%, as reported in Figure 5.13. This low level of accuracy is based on the rules identified for use by the human domain expert, but also is affected by the low quality of data.

Regardless of the data set and the imperfect deduplication rules, the accuracy reported in Figure 5.11 increased significantly when using SVM fusion to perform the classification step for the rules. Rule-level fusion achieved a far higher average accuracy (Figure 5.13) than the traditional technique, with no manual tuning of the rules and thresholds, and clue-level fusion achieved an even higher level of accuracy. This indicates that even for another rule set than those used in our experiments in Chapters 3 and 4, the use of an SVM classifier reduces the need for manual rule tuning. We would expect clue-level fusion to outperform the rule-level fusion (as shown in Figure 5.13) because both of the testing data sets contain missing data.

As shown in Figure 5.11, it is clear that as the number of missing data fields increases and as the similarity decreases between those fields that do contain data, clue-level fusion outperforms rule-level fusion, while continuing to alleviate the need for manual tuning of the user-specified thresholds. Not only does the number of false negatives drop significantly with a $CL-MM$, but the number of correctly identified duplicate records, i.e., true positives, also increases over that achieved via an $RL-MM$. 
This improvement is directly associated with the placement of fusion within the deduplication framework. These results show that on this problematic data set with 30% missing data, and even with more strict rules than those employed previously, clue-level fusion outperforms rule-level fusion because clue-level fusion is designed to be robust against missing data.

**Rule Set A: 72% Similarity, 14.03% Missing**

Figure 5.12 shows the results of Rule Set A on a separate and less problematic data set than that presented in Figure 5.11. Even though this data set contained more similar data (72% similarity), and suffered from fewer missing data fields (14.03% missing), Rule Set A produced a high number of false negatives. In fact, this high number of false negatives severely affected the sensitivity of the traditional approach. Looking at the center result chart in Figure 5.12, it is apparent that a trained SVM applied at the rule-level improved the sensitivity of the overall system dramatically, when compared to the traditional non-fused deduplication results. Again, this indicates that using an SVM classifier reduces the need for manual rule tuning.

However, when compared with the sensitivity gained via clue-level multi-SVM fusion, the efficacy of rule-level fusion is clearly not as good as that of clue-level fusion. For this data set (shown in Figure 5.12), clue-level fusion only identified an average of 299.4 false negatives, and successfully identified 3344.6 true positives, on average. Rule-level fusion, on the other hand, incorrectly identified 1447.2 false negatives and only 2196.8 true positives, on average. Again, this indicates that as the amount of missing data increases, clue-level fusion produces higher sensitivity, because it fuses around missing data. This distinct drop in accuracy when identifying which records are
Rule Set B

We now present the deduplication results achieved by applying Rule Set B to the two FEBRL-generated data sets shown in Table 5.1. Rule Set B, less strict (more lenient) than both the original rule set (used in Chapter 3 and Chapter 4) and Rule Set A (presented above), achieved higher sensitivity than the other rule sets when performing traditional non-fused deduplication. This increase in sensitivity, however, comes at the expense of specificity. See Figures 5.14-5.16 for the experimental results recorded for Rule Set B.

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Rule-Level Fusion</th>
<th>Clue-Level Fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive</td>
<td>588.2</td>
<td>2006.4</td>
<td>3050</td>
</tr>
<tr>
<td>False Positive</td>
<td>11059.2</td>
<td>382830.2</td>
<td>373071.5</td>
</tr>
<tr>
<td>False Negative</td>
<td>3007.8</td>
<td>1589.6</td>
<td>546</td>
</tr>
<tr>
<td>True Negative</td>
<td>1484344.8</td>
<td>1112573.8</td>
<td>1122332.5</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>0.1636</td>
<td>0.5580</td>
<td>0.8482</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.9926</td>
<td>0.7440</td>
<td>0.7505</td>
</tr>
</tbody>
</table>

Figure 5.14. Deduplication results obtained by Rule Set B with 30% missing data and 54.38% data set similarity. This data set contains a total of 3596 duplicates and 1495404 non-duplicates (for all 1000 probe and 1500 gallery records).
Figure 5.15. Deduplication results obtained by Rule Set B with 14% missing data and 72% data set similarity. This data set contains a total of 3644 duplicates and 1495356 non-duplicates (for all 1000 probe and 1500 gallery records).

Figure 5.16. Deduplication results of Rule Set B on both data sets: one data set containing 30% missing data (with 52.38% similarity), and a second data set with 14.03% missing data (with 72% similarity).

For both data sets described in Table 5.1, Rule Set B identified far more false positives than did Rule Set A when performing traditional non-fused deduplication.

Consider the results shown in Figure 5.11 against those shown in Figure 5.14. With 30% of the data missing, Rule Set A averaged 6947 false positives when performing traditional deduplication, but Rule Set B identified 11059.2 false positives, on average.

These high levels of false positives go hand-in-hand with the number of true positives
identified. Rule Set B is more lenient than Rule Set A and identifies more true positives at the expense of additional false negatives. With traditional deduplication, the non-tuned rules in both Rule Set A and Rule Set B fail to correctly identify a high number of duplicate records, and therefore result in low sensitivity values for the two FEBRL-generated data sets shown in Table 5.1.

Even though the number of correctly identified true duplicates in both of these data sets is low when using traditional deduplication, Rule Set B still found more true positives than Rule Set A (see Figures 5.11-5.16). In fact, Rule Set B identified about twice as many true positives as Rule Set A did, on the same data sets. This lends credence to the concern we presented previously: no rule set is perfect and only performs as well as the human domain expert understands the problem space and the current data set.

Regardless of this limitation, SVM fusion still improved the efficacy of Rule Set B. Figures 5.14-5.16 show that for Rule Set B, clue-level fusion consistently outperforms both traditional deduplication and rule-level fusion, in terms of sensitivity. Recall that we trained our SVMs for sensitivity, because of the type of cross-validation accuracy test that we performed. Therefore, we expect the SVM classifier to increase the sensitivity of the deduplication results. When applied to each data set, the results recorded for Rule Set B highlight this improvement quite well, as shown in Figure 5.16. Consider the improvements seen within Rule Set B between the two data sets. With 30% missing data (58.34% similarity), the $CL-MM$ outperforms the $RL-MM$ by 29.02%, and with only 14.03% missing data (72.00% similarity), while the $CL-MM$ outperformed the $RL-MM$ by 42.08%. Thus, as shown in Figure 5.16, even with imperfect rules and clues,
clue-level fusion outperforms rule-level fusion on missing data due to the clue-level fusion’s ability to fuse around missing data, allowing this approach to more accurately identify duplicate records in the presence of missing data.

**Discussion of Additional Experimental Results**

These additional experiments further prove the efficacy of the fused deduplication approaches presented in Chapters 3 and 4. In addition to the results reported in Chapters 3 and 4, Figures 5.11-5.16 show that rule-level fusion and clue-level fusion prove more effective than traditional non-fused deduplication, without the need for manual tuning. Thus, the experiments reported both here in Chapter 5 and in the previous chapters, empirically show that employing SVM fusion reduces the need for this potentially time-consuming process of refining and hand tuning the rule- and clue-level thresholds.

On several data sets and with three different rule sets, our clue-level fusion technique proved more effective in identifying duplicate records than did rule-level fusion, and both fusion techniques proved more effective than the traditional approach. Each time, best-fit multi-SVM applied at the clue-level produced the highest average sensitivity, further supporting Chapter 4’s hypothesis that clue-level best-fit multi-SVM fusion is robust against missing data while reducing the need for iterative, manual tuning of the rules.

**Rule-Level Fusion Versus Clue-Level Best-Fit Multi-SVM Fusion**

As previously presented, rule-level fusion takes the results of the individual rules within a deduplication system and fuses those results into one overall system-level match-score, specifying *match/no match*. This higher-level use of fusion is applicable to
any rule-based deduplication system, regardless of what the rules are, or what types of data these rules compare, as long as the resulting value returned from the individual rules is accessible and can be scaled appropriately (within the range \([0, 1]\)). Whether this scaling of the results happens within the rule or after the rules return their results is unimportant as long as scaling of the rule-level output values occurs before the rule-level fusion engine receives them as inputs. This could all be done without the need to change any of the existing rules. Thus, a rule-level fusion engine \(RL-MM\) requires no knowledge of what types of data are being compared within each rule, just that the individual rule-generated results need to be fused.

Performed by a bank of best-fit SVMs, clue-level fusion is not as transparent as rule-level fusion. This approach builds and trains the bank of SVMs based upon specific data fields, i.e., clues. Employing this type of fusion within another rule-based deduplication system would require knowledge of the specific clues used by the rules, and the data fields they leverage. Not only does this approach require insight and access to the clue set, i.e., feature set, used by each rule in the rule set, but the implementation complexity is much higher. As presented in Chapter 4, the \(CL-MM\) technique requires a trained SVM for every unique combination of clues. The set of rule/clues used in the experiments of Chapter 4, as well as here in Chapter 5, make use of a representative set of clues generated and defined for the GJXDM schema [22]. It is not difficult to see that as the number of salient and useful features within a schema increases, the number of possible clue combinations grows accordingly. Even though the experimental results for the \(RL-MM\) and the \(CL-MM\) show that the \(CL-MM\) is more accurate in identifying duplicate records (particularly with missing field data), the implementation details of
Chapter 4 make it clear that the required training is more complex than that required for the $RL - MM$.

It is important to highlight, however, that due to the robust nature of this $CL - MM$ technique, specifically in the presence of missing data, this technique moves rule-based deduplication further into the forefront of the state of the art than does the $RL - MM$ technique.

Overview of Other Matchers and Discussion

As briefly discussed in Chapter 2, several string-matching techniques exist, as well as differing approaches to record classification. The literature is, however, quite lean in regards to determining new approaches to improving the efficacy of rule-based deduplication. Some recent work, however, is directly applicable to rule-based deduplication. As mentioned in Chapter 2, Chaudhuri et al. showed that in rule-based deduplication, distinct and unique thresholds performed better than setting one, overall global threshold [15] to determine classification, hence our use of (intermediate) clue- and rule-level match-scores. Further, the ongoing research in applying support vector machines to perform match/no match classification in the realm of database deduplication helps provide further foundational work for our approach [18, 19]. However, despite this previous work, rule-based deduplication continues to suffer from the same issue: how to reduce the potentially expensive process of manually tuning the clue- and rule-thresholds that identify duplicate records. None of these previous techniques addresses this tuning problem. The identification of effective thresholds for ad hoc deduplication is a known problem, regardless of the entities represented in the database.
As discussed in Chapter 2, there are sundry and varied techniques for detecting duplicate records. These approaches include two distinct groups: (1) principled techniques, and (2) ad hoc techniques. The primary distinction between these approaches is whether the specific technique is rooted in an established analysis method or instead is a one-off rule based only in human intuition. Examples of principled duplicate detection include methods underpinned by Bayesian inference, machine learning, etc. Examples of ad hoc duplicate detection methods include rule-based systems, thresholds, etc. Each approach has strengths and weaknesses. Elmagarmid et al. summarized that while experimental results showed that principled methods are more accurate than manually tuned ad hoc approaches, they are only applicable to data sets one to two orders of magnitude smaller than data sets that ad hoc approaches can handle. Therefore, these principled approaches are significantly less scalable than ad hoc approaches [3]. Moreover, the greater accuracy of existing principled solutions comes at a price: they require significantly much more CPU time than ad hoc methods [3]. As expected, neither category provides a global best-fit solution to any and all data sets or problem domains. Our techniques attempt to bridge this gap, pushing the state-of-the-art forward. Specifically, when considering ways to reduce the cost of manual tuning, employing an SVM at the rule- or clue-level proves useful and the experimental results promising. Our fusion techniques show that using SVM fusion at least reduces the effort required to fine tune rule- and clue-level thresholds and that clue-level fusion improves deduplication results with missing data. This is a current, and until now, unresolved issue.
CHAPTER 6

CONCLUSIONS

Rule-based deduplication is effective and scalable [3]. Traditionally, this technique requires that the rules used to identify matches in a data set be tuned (or improved) through a manual, iterative process. Even though effective, this manual tuning can be intensive and time-consuming, making the deployment of these systems difficult in practice [3]. More importantly, in the presence of missing data, the accuracy achieved by rule-based deduplication systems degrades, and missing data is prevalent in real-world data sets [3, 7, 10, 20, 21]. These two critical problems affect the efficacy and use of rule-based deduplication techniques.

 Appropriately training a rule-based deduplication system using rule-level SVM fusion, and applying it to fuse individual rule results into one overall match-score, effectively reduces the need to manually tune the rules used. Rule-level SVM fusion achieves a high level of accuracy without any manual tuning of the rules and thresholds.

 In the presence of missing data, a best-fit multi-SVM approach [4] applied at the clue-level, instead of applying a single SVM at the rule-level, is effective and accurate when identifying matching data records, resulting in a higher average accuracy than by implementing a traditional non-fused deduplication rule-based deduplication engine or via rule-level fusion. Training and using a bank of SVMs, in which each SVM works on a subset of the clues used by a rule to identify matching records, provides a robust technique to identify matches when some of the identifying features are missing from the data set. Not only does this technique prove accurate in identifying matching records
when identifying data is missing, but it does not require any manual tuning of the rules. The employed SVMs learn the appropriate values necessary to deduplicate the data set in question, without any tuning by a human domain expert [3], thus satisfying the need to remove the cost of manually tuning expert-defined rules, in addition to proving accurate when missing data is prevalent [3, 4, 7, 20, 21].

Implications

Currently, the state-of-the-art in data deduplication consists of using a principled, probabilistic model, such as a Bayesian net, to perform data deduplication [3, 8, 10]. This approach requires pre-labeled training pairs to train a system to perform data deduplication. In part, this leaning in the community to use probabilistic models is due to the difficulties of deploying a rule-based deduplication engine, based on a domain expert's defined rules [3]. Even though this type of approach is effective at developing and/or generating the matching rules used in a rule-based deduplication systems, once created these rules have consistently required that a human expert manually tune them. Generating the rules by hand, or via training data fed into a learning-based system for semi-automatic rule creation, is irrelevant, since the rules still require manually tuning [3]. This type of system, i.e., one requiring expensive manual tuning, is admittedly difficult to deploy. Of late, more researchers have looked into using probabilistic matching methods instead of using rule-based systems due precisely to the person-hour cost of tuning the matching rules. Replacing the expensive manual process of iteratively tuning the matching rules with SVM fusion moves the state-of-the-art forward. The cost incurred in training an SVM is minimal, simply requiring test data deemed useful and
appropriate in identifying matching and overlapping records in a data set. After
generating appropriate training examples, selecting the training examples to use and
actually training the SVM is a semi-automated process.

Further, missing data is a current and critical issue in all types of data
deduplication [3, 7, 10, 20, 21]. Missing data increases the risk that duplicate records
will not be correctly identified, thus either relegating these partially identified records to a
manual clerical review category or causing the deduplication system to completely ignore
these records, resulting in a duplicated record set that is less than optimal. According to
the state-of-the-art, this clerical review category requires that a human review each record
set contained therein, to identify whether these records are really matches or not. This
reintroduces a time-consuming and costly iterative process, which automated
deduplication systems are trying to remove or reduce in the first place. When
implemented via a best-fit multi-SVM approach, clue-level fusion removes not only the
need to manually tune the rules and associated thresholds, but the empirical results show
that the technique is highly accurate in identifying matching and duplicate records,
reducing the need to perform the costly clerical review process.

Summary

There are many ways to perform database deduplication, ranging from
probabilistic matching models to distance-based techniques to unsupervised learning [3].
Among these is the rule-based expert system approach. Unfortunately, even with initial
rules well-defined by a human expert, this type of system suffers from the need to
perform multiple iterations of manual tuning on the rules, the thresholds employed, and
the types and values of clues used in identifying matching records, all in an attempt
improve the efficacy of these expert defined rules. Regardless of how well defined a set
of rules might be, in the presence of missing data, the efficacy of rule-based
deduplication approaches decrease.

Both a single SVM and a best-fit multi-SVM approach employed at the rule-level
and at the clue-level, respectively, make use of the expertly defined rules used in rule-
based deduplication, but through two different approaches and implementations, to
successfully identifying matching records. Rule-level fusion is performed via a single
SVM and successfully learns to classify new input records as match/no-match without the
need to manually tune the expertly defined rules. Best-fit multi-SVM fusion performs
fusion on the clues contained within each rule. This multi-SVM technique allows the
system to specify which clue-level SVM should be applied by choosing the appropriately
trained SVM from the predefined bank, depending on which identifying fields exist in the
data set.

In both applications of SVM fusion (either a single SVM employed at the rule-
level, or a best-fit multi-SVM employed at the more granular clue-level), applying fusion
to rule-based deduplication achieves high average accuracy when identifying duplicate
records, without the need to perform time-intensive manual tuning. Further, best-fit
multi-SVM fusion applied at the more granular clue-level rather than at the rule-level,
provides an effective mechanism to deal with the real-world problem of missing or
unavailable information, by maintaining a high average accuracy even when there are
missing data values, thus proving robust to missing data.
Future Work

We applied our fusion techniques at different points in the deduplication process. Our clue-level fusion was applied within a single rule, i.e., at a more granular level, and our rule-level fusion was applied to the overall system of rules. These techniques operate independently of each other; rule-level fusion effectively and accurately performed deduplication without considering clue-level fusion, and vice-versa.

Even though independent, our two fusion techniques could be used in the same deduplication system, in sequence, to create one integrated system. Best-fit multi-SVM fusion could be applied at the clue-level to compensate for missing data, and then rule-level fusion could be applied to tune the rules in the context of the overall system. Because our two techniques improve different pieces of the deduplication process, and do not depend on each other, they could be used together. The tradeoffs of time/outcome would require minimal implementation time for an expected increase in accuracy.

Rule-based systems are dependent on a human expert to define logical rules to identify matching data records [3]. However, Bayesian techniques can be used to automatically generate rules from a data set or schema [3, 8, 10]. Unfortunately, these rules still require manual tuning to perform effectively. In practice, even automatically generated rules are manually tuned so that the rules perform well on unseen records, i.e., those records not used as training data. It is unlikely that any initial set of rules will fully capture the entire problem domain and all possible record variations [3]. Learning-based fusion has yet to be applied to automatically generated deduplication rules in an effort to tune them; therefore, our fused deduplication techniques could be applied to these rules, thus automating even more of the deduplication process.
Exploring the use of *alternative machine learning algorithms* for performing clue-level fusion, in the place of multi-SVM fusion, could be useful in identifying how effective a bank of SVMs is when compared with other approaches in performing the deduplication task in the presence of missing data.

As can be seen in Figures 4.22, 5.13, and 5.16, clue-level fusion significantly outperforms rule-level fusion when approximately 15% or more of the record data is missing. However, as noted previously, clue-level fusion requires access to the individual clues that comprise each deduplication rule. Rule-level fusion, on the other hand, does not require as much knowledge of and integration with the existing deduplication system. A thorough *cost analysis* of the performance and cost of the two algorithms presented in this work would be beneficial for determining when to utilize each, and to which data sets these two solutions should be applied.
BIBLIOGRAPHY


