


Spring 5-3-2016

Identifying Relationships between Scientific Datasets

Abdussalam Alawini
Portland State University

Let us know how access to this document benefits you.

Follow this and additional works at: http://pdxscholar.library.pdx.edu/open_access_etds

 Part of the [Databases and Information Systems Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

Recommended Citation

Alawini, Abdussalam, "Identifying Relationships between Scientific Datasets" (2016). *Dissertations and Theses*. Paper 2922.

10.15760/etd.2918

This Dissertation is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. For more information, please contact pdxscholar@pdx.edu.

Identifying Relationships between Scientific Datasets

by

Abdussalam Alawini

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

Doctor of Philosophy
in
Computer Science

Dissertation Committee:

David Maier, Chair

Kristin Tufte

Melanie Mitchell

Tugrul Daim

Portland State University

2016

© 2016 Abdussalam Alawini

ABSTRACT

Scientific datasets associated with a research project can proliferate over time as a result of activities such as sharing datasets among collaborators, extending existing datasets with new measurements, and extracting subsets of data for analysis. As such datasets begin to accumulate, it becomes increasingly difficult for a scientist to keep track of their derivation history, which complicates data sharing, provenance tracking, and scientific reproducibility. Understanding what relationships exist between datasets can help scientists recall their original derivation history. For instance, if dataset A is contained in dataset B , then the connection between A and B could be that A was extended to create B .

We present a relationship-identification methodology as a solution to this problem. To examine the feasibility of our approach, we articulated a set of relevant relationships, developed algorithms for efficient discovery of these relationships, and organized these algorithms into a new system called ReConnect to assist scientists in relationship discovery. We also evaluated existing alternative approaches that rely on flagging differences between two spreadsheets and found that they were impractical for many relationship-discovery tasks. Additionally, we conducted a user study, which showed that relationships do occur in real-world spreadsheets, and that ReConnect can improve scientists' ability to detect such relationships between datasets.

The promising results of ReConnect's evaluation encouraged us to explore a

more automated approach for relationship discovery. In this dissertation, we introduce an automated end-to-end prototype system, ReDiscover, that identifies, from a collection of datasets, the pairs that are most likely related, and the relationship between them. Our experimental results demonstrate the overall effectiveness of ReDiscover in predicting relationships in a scientist's or a small group of researchers' collections of datasets, and the sensitivity of the overall system to the performance of its various components.

DEDICATION

To my mother and father, my endless source of inspiration and dedication.

To my wonderful wife, without whose support I would not have finished my doctoral studies.

To the joy of my life, my lovely kids, for making this journey more fun than it really is.

ACKNOWLEDGMENTS

Many thanks to my adviser, Professor David Maier, for his support and help throughout my studies. He is a fun person to work with. I definitely learned a lot from his experience, dedication, and passion for exploration.

I would also like to thank my co-adviser, Dr. Kristin Tufte, for always pushing my limits, and making sure that Dave and I were always on track.

I have had much support from my committee members. Special thanks to Professor Melanie Mitchell and her students for helping with the machine learning part of my research. Many thanks to Professor Tugrul Daim for his valuable feedback and guidance.

Thanks to Rashmi Nandikur for helping with the implementation and testing of ReDiscover.

I would like to thank Dr. Bill Howe of the University of Washington for his collaboration, and for providing us with access to scientific dataset repositories. I would also like to thank Dr. Mike Cafarella of the University of Michigan and his students for their help.

Last but not least, I would like to thank my colleagues at the Portland State University Datalab, including Patrick Leyshock (alumnus), Veronika Megler (alumna), Hisham Benotman and Basem Elazzabi, for all their support and feedback. Special thanks to Scott Britell for his technical support, and for making the Datalab a fun place for work. Thanks also to Dr. Lois Delcambre for her support and encouragement.

My research was supported by the National Science Foundation (award IIS 1064685), and by the Libyan Ministry of Higher Education and Scientific Research through the Canadian Bureau for International Education (CBIE). Thanks CBIE for your commitment to the Libyan-North American Scholarship Program, and for your excellent support and services.

Table of Contents

Abstract	i
Dedication	ii
Acknowledgments	iv
List of Tables	x
List of Figures	xi
Chapter 1: Introduction	1
Chapter 2: Motivation, Overview and Background	8
2.1 Motivation	8
2.1.1 A Motivating Example	9
2.2 Conceptual Basis for Relationship Identification	11
2.2.1 Activities, Connections and Relationships	11
2.2.2 Relationship Definitions	14
Ordered Relationships.	16
2.3 Background	19
2.3.1 Relationship Testing	20
Data Profiling	20
Schema Matching	21
2.3.2 Relationship Prediction	21
Data-Column Extraction	22
Automated Column Matching and Relationship Prediction	22
Summarizing Categorical Columns	23
Chapter 3: Relationship Testing	26
3.1 ReConnect’s Architecture	26

3.1.1	Converting Spreadsheets to Database Tables	26
3.1.2	Column Correspondence	28
3.1.3	Relationship Identification	31
3.2	User Study and Evaluation	37
3.2.1	User Study	38
3.2.2	Methodological Evaluation	43
3.2.3	Results	45
Chapter 4: Relationship Prediction		50
4.1	Challenges	50
4.1.1	Automating the Extraction of Tabular Datasets from Spreadsheets	51
	Wide Variety of Dataset Layouts in Spreadsheets	51
	Non-data Text	52
	Producing Training Data for Cell Labeling	52
4.1.2	Scaling to Many Datasets and Columns	52
4.1.3	The Need for Sophisticated Column Matching	53
	Inferring the Type of a Column	53
	Summarizing Categorical Columns	53
4.2	A Description of the Rediscover System	54
4.2.1	Label Cells	55
4.2.2	Extract Columns	60
4.2.3	Compute Column Summaries	63
4.2.4	Match Columns	64
	Computing Set Similarity using Bloom Filters	67
4.2.5	Predict Relationships	71
	Predict Relationships Process	72
	Computing Relationship Features	72
	Predict Relationships Results	75
Chapter 5: Experimental Evaluation		77
5.1	Preliminary Evaluation	77
5.1.1	Methodology	78
5.1.2	Results	80
5.1.3	Discussion	82
5.2	Label Cells Experiment	84

5.2.1	Methodology	84
5.2.2	Results and Discussion	85
5.3	Predict-Relationships Experiment	86
5.3.1	Methodology	87
5.3.2	Results	89
5.3.3	Discussion	93
Chapter 6:	Related Work	96
6.1	Scientific Data Management	96
6.2	Relationship Testing	100
6.2.1	Similar tools	100
6.2.2	Relevant Techniques	101
6.3	Relationship Prediction	102
6.3.1	Tabular Dataset Extraction	103
6.3.2	Automated Schema Matching Techniques	107
6.3.3	Similarity Detection using Bloom Filter	111
Chapter 7:	Future Work, Further Applications, and Conclusions	113
7.1	Future Work	113
7.1.1	Multi-Dataset Connection Identification	114
7.1.2	Scaling to Large Dataset Collections	115
7.1.3	Improving ReDiscover’s Prediction Performance	118
	Evaluating Relationship-Prediction Features	118
	Conducting Further User Studies	119
	Improving Column Matching	119
	Exploiting Other Spreadsheet Features	120
7.1.4	Integrating ReConnet with ReDiscover	121
7.2	Further Applications	122
7.2.1	Applications with Similar Relationships	123
7.2.2	Applications with New Relationships	123
7.2.3	Applications with New Relationships Requiring New Features	125
7.2.4	Applications with Different Data Formats	126
7.3	Conclusion	128
References	131
Appendix A:	Cell Features	141

Appendix B: ReDiscover’s Column Summaries 145
Appendix C: Human-Based Approach Features 148

List of Tables

Table 2.1	An example of the activities that Jennifer’s collaborators may have performed on spreadsheets <i>A</i> , <i>B</i> , <i>C</i> , and <i>D</i>	12
Table 2.2	Description of some of the relationships that the relationship-identification methodology identifies.	15
Table 2.3	Description of ordered relationships.	18
Table 3.1	The methodological evaluation results	46
Table 4.1	Description of spreadsheet metadata that ReDiscover collects.	64
Table 5.1	Label Cells Performance.	85

List of Figures

Figure 2.1	An example of three related spreadsheets: Spreadsheet <i>A</i> is <i>row-contained</i> in spreadsheet <i>B</i> , and spreadsheets <i>B</i> and <i>C</i> are <i>complementary</i>	9
Figure 2.2	An example of <i>augmentation</i> and <i>sub-containment</i> relationships	16
Figure 2.3	An example of <i>prefix</i> and <i>subsequence</i> relationships	17
Figure 3.1	ReConnect’s architecture: The process <i>Upload Dataset</i> converts a spreadsheet into a database table, and the process <i>Identify Relationships</i> detects relationships between pairs of spreadsheets.	27
Figure 3.2	Dataset selection for spreadsheet <i>B</i> : Jennifer can specify column and row ranges and the index of the row that contains the column names of her dataset.	28
Figure 3.3	<i>Column Correspondence</i> process for Spreadsheets <i>A</i> and <i>B</i> : ReConnect depends on a user’s feedback to identify column correspondences accurately.	29
Figure 3.4	The initial column correspondence between datasets <i>A</i> and <i>B</i>	30
Figure 3.5	The <i>Identify Relationships</i> Process: ReConnect analyzes column correspondences, column statistics, and the set of common rows of two tables to check if they match the conditions for a relationship	32
Figure 3.6	Relationship Identification Overview: Classifying relationships based on column correspondences helps ReConnect limit the relationships to investigate.	33
Figure 3.7	An example of two equal spreadsheets with reordered rows and columns: Change-inference tools report that spreadsheets <i>B</i> and <i>B'</i> are not equal (identical). ReConnect reports that they are <i>equal</i> regardless of their row and column order.	47

Figure 3.8	SheetDiff results of comparing spreadsheets B and B' (shown in Figure 3.7). Yellow is used for changed cells, blue for added columns and rows, and red for deleted columns and rows.	48
Figure 3.9	Micorsoft Inquire results of comparing spreadsheets B and B' (shown in Figure 3.7).	48
Figure 4.1	The main processes in ReDiscover	54
Figure 4.2	<i>The Label Cells</i> Process	56
Figure 4.3	Example Label Cells results	59
Figure 4.4	Extract Columns	60
Figure 4.5	Column Layout Examples	61
Figure 4.6	An example of column data-type inconsistency in spreadsheets	62
Figure 4.7	Example column summaries	63
Figure 4.8	An example of two column summaries and their similarity vector	66
Figure 4.9	The Match-Column process	67
Figure 4.10	An example of enhancing the column correspondence of datasets D and E	69
Figure 4.11	Example results of the <i>Predict Relationships</i> algorithm . . .	75
Figure 5.1	Relationship prediction performance results of ReDiscover and ReDiscoverMCS	78
Figure 5.2	ReDiscover's Average Prediction Score	82
Figure 5.3	Average interpolated precision at standard recall levels, with average prediction scores across all relationships	90
Figure 5.4	Interpolated precision at standard recall levels, with prediction scores.	92
Figure 5.5	The results of ReDiscover's prediction score evaluation (Part 3).	94
Figure 6.1	An example of data-presentation spreadsheet: Resident population, by age, sex, race, and Hispanic origin: United States, selected years 1950-2009 (A partial picture of the original spreadsheet). Source: [14].	106
Figure 6.2	The classification of ReDiscover's schema-matching approach based on Rahm et al.'s taxonomy (Highlighting was added). Source: [60].	108
Figure 7.1	An example of a graph of predicted relationships in a collection.	115

Figure 7.2	A possible approach for using blocking techniques to scale ReDiscover.	117
Figure 7.3	An example of two versions of a dataset about home maintenance costs, with a <i>near-match</i> relationship. The values of <i>Cost</i> column of dataset Maintenance_V1 has been rounded to the nearest dollar in Maintenance_V2 dataset.	124
Figure 7.4	An example of using Bloom filters for computing indicative features for the <i>bag-row-containment</i> relationship.	125
Figure 7.5	A sample spreadsheet with vertical and horizontal labels and subtotals.	127

CHAPTER 1: INTRODUCTION

Large amounts of scientific data currently exist as dataset files outside of database management systems (DBMSs). As scientists perform various activities with these datasets—such as combining data split across several files, filtering or rearranging data to aid their analyses, and sharing datasets with collaborators and receiving versions with modifications—the size and the number of dataset files related to a research project increases. In such cases, scientists can lose track of the connections between their files and it becomes increasingly difficult for them to answer questions such as: Which dataset is the most complete? Which two versions of datasets go together? Are there any overlapping datasets? Are there any duplicate datasets? Not having ready answers to these questions can be an impediment for scientists to share their data or move it to a DBMS, and can add to the time that they spend managing their datasets rather than analyzing their data [74].

We believe that determining relationships, such as *row-containment*, *complementation*, *duplicate*, and *augmentation*, between datasets can help answer such questions, and is the focus of this work. For example, assume that a researcher is trying to determine the most complete version among three datasets stored in spreadsheets X , Y , and Z , to share with her collaborators. Knowing that datasets X and Y are row-contained in Z can help her determine that the most complete version is Z , as it contains the rows of both datasets X and Y . As another example, suppose that a scientist performed bacteriological analysis on a set of water samples and saved it as spreadsheet A , and then created a new spreadsheet B to

store her chemical analysis on the same set of water samples. A few months later, she wants to decide whether to upload A , B or both to a shared archive. The classification of the relationship between these spreadsheets as *complementation* can help her determine that the connection between them is that the unmatched columns provide complementary information about the matching water samples in A and B . Consequently, she may choose to upload the two spreadsheets together.

Version history could give some hints about connections between datasets, but that by itself is not enough for the following two reasons. First, version control systems (VCS) cannot always track versions across all scientists' activities on spreadsheets. For instance, when a scientist sends a spreadsheet to a collaborator, the changes made by her collaborator would not be tracked by the VCS. Second, relationships could arise between datasets without direct derivation. For instance, the complementation relationship we just mentioned in the example above could arise not necessarily from having dataset A derived from B , or vice versa. Thus, we sometimes need to determine relationships between datasets by looking at the data stored in them.

In this work we focus on tabular datasets because much of the file-based scientific data are organized as tables, such as spreadsheets, instrument output, and sensor logs. Moreover, in some cases, ordering of these tabular datasets is significant and can help in determining how datasets are connected. For instance, as ordered tables are manipulated and transformed for analyses, different kinds of ordered relationships, such as *infix*, *sub-sequence*, and *reordered rows*, can result between source data, intermediate results, and derived datasets. We need an approach that can handle relationships involving order.

Since we believe the relationship-identification problem is important, we wanted to develop an approach to it. Our research had two phases. In the first phase, we focused on the viability of our relationship-identification approach. We wanted to know whether relationships arose in practice, whether we could effectively identify

them, and, if so, are relationships useful for scientists in identifying the original connections between their datasets? Thus, we built a prototype tool, ReConnect, that helps a scientist interactively determine connections between two datasets. The results from ReConnect evaluation were promising, and we were encouraged by what we observed. However, the interactive, pairwise approach was not suitable for a scientist with more than few datasets. So, in the second phase, we wanted to make relationship identification practical at the scale of dataset collections. To support that case, we had to go beyond pairwise relationship testing, and develop a more automated approach that reduces user involvement as much as possible. To that end, we developed ReDiscover, an end-to-end prototype system that helps scientists determine which pairs to examine for relationships from a collection of datasets.

More specifically, in the first phase, we articulated a set of relevant relationships (Section 2.2.1) and developed a methodology for identifying these relationships between pairs of spreadsheets. This methodology first extracts tabular datasets from spreadsheets. Then it extracts column features to suggest a column correspondence and also to suggest potential relationships between datasets. Finally, it establishes correspondences between sets of columns in order to generate SQL queries to examine column data to confirm suggested relationships. To examine the viability of our methodology, we first implemented and evaluated ReConnect (Section 3.1), an interactive tool that implements the methodology. Second, we conducted a user study where we talked to scientists to both validate our intuition about relationships and to examine ReConnect’s usefulness in determining meaningful relationships (Section 3.2.1). This study confirmed that the proliferation of scientific datasets can be a significant problem, and that scientists often struggle to select and decide how to work with their datasets, especially when working with collections of datasets where they are uncertain about the connections.

The work on ReConnect established the viability of our relationship-identification

approach as it verified the following two hypotheses. First, *finding relationships matters* because 1) relationships do actually occur in practice, 2) knowledge of them can aid scientists in determining original connections between their datasets, and 3) scientists cannot easily check them manually in spreadsheets of even modest size. Second, *we can at least partially automate relationship identification and testing*, as is demonstrated by ReConnect. However, it is tedious for scientists to apply ReConnect to a collection of, say, 50 datasets, as it would involve looking at 1200+ dataset pairs. Thus, we now needed to develop an approach that is practical for a scientist to apply across whole collections of his or her datasets.

In the second phase, we examined the applicability of our approach by developing a relationship-prediction methodology for suggesting the pairs that are likely related, and the relationship between them, in a collection of datasets. Our methodology applies a conditional probabilistic model, namely Conditional Random Fields (CRFs, Section 4.2.1), to automate data-column extraction (Section 4.2.2), computes fast approximate column summaries using data profiling and Bloom filters (Sections 4.2.3, and 4.2.4), and applies Support Vector Machines (SVMs) to automate column matching (Section 4.2.4) and to predict relationships (Section 4.2.5) between dataset pairs. Specifically, we make the following contributions.

- *Development of an approach for automatically extracting tabular datasets from spreadsheets.* We developed a new approach that applies CRFs to identify spreadsheet cells that are part of a data column, group these cells as columns, and identify data types for resulting columns. We show that this approach is capable of extracting datasets in real-life and synthetic spreadsheets with reasonable accuracy.
- *Design and implementation of a summarization technique for categorical column.* We designed a method based on Bloom filters that converts each categorical column into a fixed-size bit-vector. Using these vectors, the

relationship-prediction methodology computes similarity between categorical data while preserving our scalable architecture, which avoids comparing individual data values and bounds the amount of effort required to derive joint features.

- *Development of a new technique, based on Bloom filters, for computing indicative relationship features.* We developed a new technique that uses Bloom filters for computing indicative features for certain relationships, such as *duplicate*, *containment*, *prefix*, and *suffix*.
- *Prototype implementation of a scalable system for predicting relationships (called ReDiscover).* To evaluate our methodology, we developed ReDiscover, an end-to-end prototype prediction system that takes a collection of spreadsheets as input and produces a list of pairs of datasets and the predicted relationship between them, if any.
- *Experimental validation of the relationship-prediction approach.* We evaluate the ReDiscover approach, demonstrating the effectiveness of our relationship-prediction methodology. We also evaluate our dataset-extraction technique, as the accuracy of ReDiscover’s predictions depends heavily on the accuracy of the dataset-extraction steps.

While relationships can occur in any kind of datasets, this work focuses on spreadsheets for the following reasons. First, spreadsheets are widely used for storing tabular datasets across disciplines, whereas other scientific dataset formats, such as Flexible Image Transport System (FITS) [73], Network Common Data Form (netCDF) [63] and Hierarchical Data Format (HDF) [40], are more domain specific.

Second, based on the file-based datasets we examined, we noticed that many datasets can be characterized as ordered tables at the abstract level. The order of

rows and columns can provide indications of past activities, which we can help determine by identifying order-based relationships, such as *prefix* (see Section 2.2.2). Spreadsheets are one of the formats that capture dataset order. While relational tables can model order, order is not inherent in them and extra work is required to capture it.

Third, to identify relationships, our methodology relies on extracting indicative features from datasets, such as column type and value frequency. We also wanted to explore using additional kinds of information as indicators for relationships, such as spreadsheet metadata (e.g. file name, size, and author) and cell properties (e.g. text alignment, font style, and data format). The trade-off here is that some other dataset formats have stronger typing of data, which could also aid relationship identification. However, ReConnect and ReDiscover are equipped with a type extractor for spreadsheets that allows them to utilize type information in identifying relationships between datasets.

This dissertation is organized as follows: In Chapter 2, we first discuss the context and motivation for our research. Next, we discuss related background research, especially existing techniques and tools that scientists could use to manage their datasets. Finally, we introduce the theoretical concepts underlying our relationship-identification methodology, catalog a set of relationships that could help scientists discover connections between their datasets, and illustrate our methodology with an example. In Chapter 3, we introduce ReConnect, a semi-automated tool for detecting relationships between two datasets. The second section of that chapter presents an assessment of ReConnect, which involves a user study, plus an evaluation of the effectiveness of ReConnect relative to four other change-inference tools in identifying relationships between spreadsheets. Chapter 4 introduces our relationship-prediction methodology, discusses the challenges related to predicting relationships in large collections of spreadsheets and presents the architecture of ReDiscover and detailed description of its components. Chapter

5 presents our experimental evaluation of ReDiscover, which involves three experiments: 1) an assessment of the effect of the result quality of each of ReDiscover's components on the results of later stages, 2) an evaluation of our data-column extraction technique and 3) an investigation of the overall effectiveness of ReDiscover in predicting relationships between datasets. In Chapter 6, we discuss related research work, and we conclude this dissertation in Chapter 7 with a discussion of further applications for our methodology and possible future research directions.

CHAPTER 2: MOTIVATION, OVERVIEW AND BACKGROUND

In this chapter, we first discuss the motivation and context of our work in Section 2.1. Section 2.2 discusses the conceptual basis for the relationship-identification methodology, and informally defines the set of relationships we developed for our methodology. Lastly, Section 2.3 presents concepts and techniques from several research areas, including data management, machine learning, and data mining, that we used in our research.

2.1 MOTIVATION

We conducted our work in the context of SQLShare [42], a database service targeted at small groups of scientists that cannot afford to hire database experts, nor have the technical skills to effectively use relational database technology. When trying to use database systems, these scientists confront several impediments, including the installation, configuration, schema design of database systems, data loading, data cleaning, and query formulation. With SQLShare, scientists may simply upload their data, query it to receive an answer, define views to improve data usability, and share results with other SQLShare users without encountering these impediments. To help users with query formulation, SQLShare team provides a set of *starter queries*—SQL queries based on the user’s uploaded datasets.

However, we observed an additional impediment to using SQLShare in some cases. Because scientists often accumulate many datasets over the course of a project, it can be difficult for them to sort through their collections to determine which ones to upload: Which dataset is the most complete? Which two versions of datasets go together? Do two datasets overlap? In fact, several scientists have

asked our collaborators at University of Washington who developed SQLShare for help with this selection problem. Similar problems exist with other data sharing platforms such as Fusion Tables [38], Dryad [43], ICPSR [1], and emerging services related to consortia such as DataOne [55] and the Research Data Alliance [2].

The goal of our work is to overcome this additional impediment: the problem of deciding which datasets to select, and which course of action to perform on these datasets. Thus, we began looking for ways to help scientists overcome this impediment. These tasks often requires that scientists determine (or recall) the original connections between datasets. Identifying relationships between these datasets could help with this determination, and is the focus of this dissertation.

Many of these datasets exist in file-based formats, such as spreadsheets and CSVs. To the best of our knowledge, there is no tool that a scientist can use for identifying connections between such datasets in a collection. Previous research focused on managing scientific data stored in databases. In Chapter 6, we discuss this part of related research in more detail.

In the next section, we present a motivating example, which we use as a running example throughout the dissertation. This example will help better explain how identifying relationships between spreadsheets can help scientists make informed decisions on how to work with their data.

2.1.1 A Motivating Example

Spreadsheet A				Spreadsheet B				Spreadsheet C			
Site	Sample	Bottle gp	Depth	Site	Sample #	Bottle gp	Depth	Site	Sample #	temp	salinity
NH-10	1350	1-3	70.4	NH-10	1350	1-3	70.4	NH-10	1350	8.815	32.6
NH-10	1351	4-6	60.9	NH-10	1351	4-6	60.9	NH-10	1351	8.814	32.6
				NH-10	1352	7-9	28.7	NH-10	1352	8.805	32.5
				NH-10	1353	10-12	3.9	NH-10	1353	8.834	32.5

Figure 2.1: An example of three related spreadsheets: Spreadsheet *A* is *row-contained* in spreadsheet *B*, and spreadsheets *B* and *C* are *complementary*.

Jennifer, a marine scientist, collaborates with a group of colleagues on assessing the effects of climate change on the Pacific Ocean. She is responsible for managing and analyzing multiple spreadsheets that contain ocean-observation data, and wants to upload data to an online service, such as SQLShare, for sharing datasets. However, she often receives multiple versions of a spreadsheet from different collaborators. Consequently, Jennifer has to inspect spreadsheets manually to identify the best ones to upload.

Figure 2.1 shows three related spreadsheets. Spreadsheets *A* and *B* contain water sample data including site ID, sample number, sample source, and the depth at which the sample was collected. Spreadsheet *B* was created by extending Spreadsheet *A* with two additional rows (rows 4 and 5). The relationship between *A* and *B* is *row-containment* because Spreadsheet *A*'s data is contained in Spreadsheet *B*. Spreadsheet *C* contains the temperature and salinity readings for the same water samples found in Spreadsheet *B*. Spreadsheet *C* agrees with *B* on the *Site* and *Sample #* columns, but the rest of the columns are not related by a column correspondence (*Bottle gp* and *Depth* of *B* do not match *temp* and *salinity* of *C*.) The relationship between *B* and *C* is *complement*, since the unmatched columns provide complementary information about the water samples in *B* and *C*.

Jennifer needs a tool to aid her in determining that spreadsheet *B* is a more complete version of spreadsheet *A*, and consequently *B* is the one that she should upload. Additionally, if she is to upload *B*, she should also upload *C* as they complement each other. In this example, it is easy to eyeball the relationships, but it would be a tedious and error-prone task to manually determine relationships between spreadsheets with hundreds of rows and tens of columns. However, no current tools specifically target this problem.

2.2 CONCEPTUAL BASIS FOR RELATIONSHIP IDENTIFICATION

We first discuss the conceptual basis of how identifying relationships between datasets can help scientists recall their original connections. (Section 2.2.1). Then, we informally define some of the relationships that we seek to identify (Section 2.2.2).

2.2.1 Activities, Connections and Relationships

Spreadsheets (and other datasets) are produced and modified as a result of scientists' *activities*, and may proliferate as a result of some of those activities. For example, a user may combine data from multiple spreadsheets, start a new spreadsheet for each day's observations, fill in missing or null values in an existing spreadsheet, or filter or rearrange data to aid in analyses. Understanding that such activities have occurred can help the user detect how spreadsheets are *connected* and how to work with their data.

Table 2.1 shows activities that Jennifer's collaborators may have performed on spreadsheets *A*, *B*, *C*, and *D*. Some activities will produce distinct relationships, as in the case of the sort activity, while other activities can produce the same relationship, as in the case of the adding- versus selecting-rows activities. User feedback is needed to remove such ambiguity in identifying connections. We want to provide users with contributing evidence so that they can make the decision as to how their datasets connect to each other.

Of course, it may be the case that two spreadsheets were not involved in a common activity. We can also identify the *incompatible* relationship, which indicates the absence of a connection.

We observe that scientists' varied activities typically produce various *relationships* between their spreadsheets. We want to detect these relationships by analyzing the data in two spreadsheets without necessarily having knowledge of the

Table 2.1: An example of the activities that Jennifer’s collaborators may have performed on spreadsheets *A*, *B*, *C*, and *D*.

Activity	Connection	Relationship
Add two water-sample rows (1352 and 1353) to spreadsheet <i>A</i> and save it as <i>B</i> .	<i>B</i> is row-extension of <i>A</i>	<i>A</i> is row-contained in <i>B</i>
Select rows with a depth greater than 30 meters from spreadsheet <i>B</i> and save it as <i>A</i> .	<i>A</i> is a selected subset of <i>B</i>	<i>A</i> is row-contained in <i>B</i>
Store the bottle label and depth of samples 1350 to 1353 in spreadsheet <i>B</i> , and store the water temperature and salinity of the same samples in spreadsheet <i>C</i> .	Unmatched columns provide complementary information about the matching water samples in <i>B</i> and <i>C</i>	<i>B</i> and <i>C</i> are complementary
Sort spreadsheet <i>B</i> on increasing <i>Depth</i> and save it as <i>D</i> .	<i>D</i> is a reordering of <i>B</i>	<i>B</i> is row-equal to <i>D</i>

history of the activities that produced or modified these spreadsheets. As we discussed in the introduction, while version or derivation history could give some tips about how two datasets are connected, relying on such information is not enough for this task because 1) VCSs cannot always track versions across different scientists’ activities on spreadsheets, especially when sharing of datasets passes outside the tracking system, 2) relationships can arise without direct derivation, and 3)

just knowing that a dataset A is a version of B does not always indicate what activities changed B .

In this work, we focus on the kinds of activities that scientists perform when operating on scientific datasets stored in spreadsheets, and we formulate the following two hypotheses. First, determining relationships is useful for scientists, as it can help them identify connections between their datasets. Second, we can at least partially automate the process of relationship discovery.

In order to investigate these hypotheses, we built ReConnect, a tool that can help identify relationships between two datasets (Chapter 2.3.2). One of the main reasons for building this tool was to evaluate whether our first hypothesis is true before investing more effort in the development of a more automated approach. We were able to validate our first hypothesis by conducting a user study that evaluated the usefulness of ReConnect and the set of relationships it identifies (see Section 3.2.1). On the basis of these encouraging results, we developed the relationship-prediction methodology (ReDiscover) to fully automate our approach (see Chapter 3.2.3).

ReConnect determines relationships between pairs of spreadsheets by first extracting tabular datasets from them with the help of the user. Second, it extracts column features to suggest a column correspondence and also to suggest potential relationships between tabular datasets. Third, our tool establishes correspondences between sets of columns, with optional user interaction. Determining column correspondences is important for discovering relationships because our relationships are categorized based on how the columns of two datasets correspond. Fourth, based on the column correspondence and column statistics, ReConnect suggests a set of possible relationships. Lastly, a user can select, from the list of suggested relationships, which relationship to investigate, and then our tool generates SQL queries to examine the extracted datasets to validate the selected relationship. In the next section, we introduce some of the relationships that are based on scientists'

activities on their spreadsheets.

2.2.2 Relationship Definitions

We have developed a set of relationship definitions by inspecting spreadsheets that we have collected from several sources, such as the EUSES spreadsheet corpus [33] and spreadsheets provided by our scientific collaborators. Tables 2.2 and 2.3 present some of these relationships. Some relationships are special cases of others. For example, prefix, suffix, infix, and subsequence are special cases of the row-containment relationship, and reordered columns and reordered rows are special cases of the equal relationship. Consequently, relationships are not mutually exclusive, and hence identifying generic relationships (e.g., row-containment) may indicate the presence of their special cases (e.g., prefix).

Our relationships are defined based on the dataset content of the spreadsheets, and not their appearance or layout. Our approach considers data the same regardless of its position or formatting. More specifically, relationship definitions are classified based on how the columns of two spreadsheets correspond, and on the data shared between their rows. Two columns *correspond* when they are semantically related and they describe the same real-world object. A *Column correspondence* is the maximal set of corresponding column pairs between two datasets. We classify the column correspondence as *Full Correspondence*, *Sub-correspondence*, *Extension Correspondence*, and *No Correspondence*. Full correspondence is when each column in the first dataset (T1) corresponds to a column in the second dataset (T2) and vice versa; sub-correspondence is when proper subsets of the columns of T1 and T2 correspond; extension correspondence is when all of T1’s columns correspond to a proper subset of T2’s columns; and no correspondence is when there is no column in T1 that corresponds to any column in T2.

Figure 2.2 shows an example of the augmentation and sub-containment relationships. Notice that spreadsheet *A* fully corresponds to *B*, and that all of *A*’s

Table 2.2: Description of some of the relationships that the relationship-identification methodology identifies.

Relationship	Description
<i>Row-containment</i> “A is row-contained in B”	When A fully corresponds to B, and the rows of spreadsheet A are a subset of the rows of spreadsheet B.
<i>Column-containment</i> “A is column-contained in B”	When A extensionally corresponds to B, and the rows of spreadsheet A are equal to the corresponding parts of the rows of spreadsheet B.
<i>Sub-containment</i> “A sub-contained in B”	When A sub-corresponds to B, and a portion of the rows of spreadsheet A is a subset of the rows in the corresponding columns of spreadsheet B.
<i>Augmentation (Fill-in)</i> “B augments A”	When A fully corresponds to B, and all rows of spreadsheets A and B match except for particular cells, and these cells are empty or null cells in A but are filled in B.
<i>Complementation*</i>	When A sub-corresponds to B, where the rows in the sub-corresponding columns match, and the remaining columns are not related by column correspondence.
<i>Template*</i>	When A fully corresponds to B but their data rows are disjoint.
<i>Equal*</i>	When A fully corresponds to B, and row-containment holds in both direction between A and B.
<i>Incompatible*</i>	No correspondence between A and B.

* Symmetric relationships.

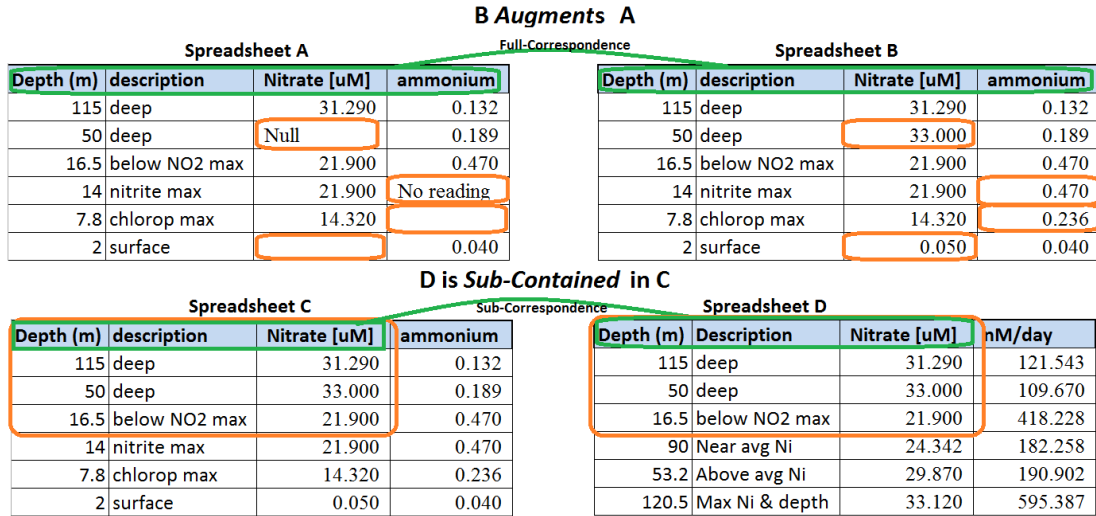


Figure 2.2: An example of *augmentation* and *sub-containment* relationships

cells with missing information, including *Null*, empty, and *No reading* values, have been filled in *B*. Thus, *B* is an *augmented* (filled-in) version of *A*. Spreadsheet *C* sub-corresponds to *D* because a subset of *C* columns corresponds to a subset of *D* columns (i.e., $C.Depth(m) \leftrightarrow D.Depth(m)$, $C.description \leftrightarrow D.Description$, and $C.Nitrate[uM] \leftrightarrow D.Nitrate[uM]$). Further, a portion of the rows of *D* is a subset of the rows in the corresponding columns of *C*. Consequently, *D* is *sub-contained* in *C*.

We considered expressing relationships with relational algebra expressions. However, we found that they are not a good match because some relationships involve preserving duplication and order, which relational algebra does not directly support.

Ordered Relationships.

The row and column order between two spreadsheets can provide useful information about the connection between them. Table 2.3 lists some of the ordered relationships. In Jennifer’s spreadsheets, when the *row-containment* relationship

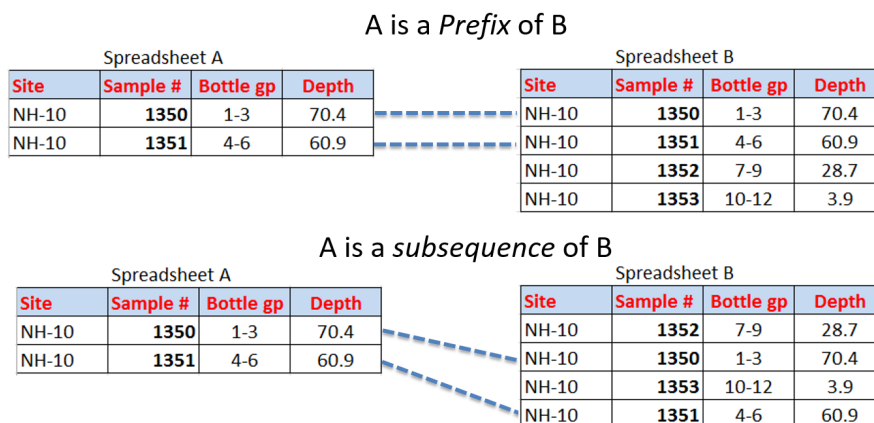


Figure 2.3: An example of *prefix* and *subsequence* relationships

holds between spreadsheet A and B , other relationships based on row order that might also hold are A is a *prefix* of B or A is a *subsequence* of B , as shown in Figure 2.3.

In the case of a *prefix* relationship, it seems more likely that A was extended to form B , but in the case of *subsequence* relationship, it appears more likely that A is a filtered version (a selection) of the rows in spreadsheet B . This example shows how detecting multiple relationships can help users understand the connection between their spreadsheets. Our intended approach should detect ordered relationships, such as *prefix*, *suffix*, *infix*, *duplicate*, *subsequence*, *reordered columns*, *reordered rows*, and *reordered columns and rows*.

As we discussed in the previous section, our relationships abstract from spreadsheet layout. Thus, the *duplicate* relationship, which is a special case of the *equal* relationship, does not mean that two spreadsheet files are identical. Datasets A and B can be duplicates while they have different column names, there are other datasets in the spreadsheets, they appear in different positions. Thus, the duplicate relationship means more than just that a spreadsheet file is a copy of another. We discuss the applicability of our approach to spreadsheets for other domains below.

Table 2.3: Description of ordered relationships.

Relationship	Description
<i>Prefix</i> “ A is a prefix of B ”	When A is row-contained in B and the rows of A appear consecutively at the beginning of B in original order.
<i>Infix</i> “ A is an infix of B ”	When A is row-contained in B and the rows of A appear consecutively at the middle of B in original order.
<i>Suffix</i> “ A is a suffix of B ”	When A is row-contained in B and the rows of A appear at the end of B with the same original order.
<i>Subsequence</i> “ A is a subsequence of B ”	When A is row-contained in B and the rows of A appear in B in the same relative order.
<i>Duplicate*</i>	When A and B are equal and have the same row and column order.
<i>Reordered Columns*</i>	A and B are equal with different column order.
<i>Reordered Rows*</i>	A and B are equal with different row order.
<i>Reordered Columns and Rows*</i>	A and B are equal with different column and row orders.

* Symmetric relationships.

Is our approach appropriate for spreadsheets in general?

Our focus is on spreadsheets that contain scientific datasets, but spreadsheets are also used for many other kinds of tabular data: sales reports, budgets, gradesheets, etc. A natural question is whether our methods work for this data. We can certainly detect instances of the current relationships between data in such spreadsheets. But the kinds of relationships that exist and their likelihood might be different for other data sources than what is common for spreadsheets for scientific datasets.

For instance, consider a spreadsheet containing a project budget, which has individual entries modified many times—a pattern we do not see much with append-mostly scientific datasets. Our relationship-identification methodology currently does not detect a relationship that specifically corresponds to this activity, or other activities uncommon with scientific datasets. In order for our methodology to identify such relationships, we would need to build new routines that can handle these new kinds of relationships.

In the next section, we discuss concepts from the literature that we made use of in developing our approaches for relationship testing (ReConnect) and prediction (ReDiscover).

2.3 BACKGROUND

In our research we use a number of techniques from several areas; we will discuss these techniques and how we use them below. In Section 2.3.1 we address the two main database techniques used in ReConnect, data profiling and schema matching. In Section 2.3.2 we introduce and briefly discuss how ReDiscover uses Conditional Random Fields, Support Vector Machines, and Bloom filters in predicting relationships between datasets in a collection.

2.3.1 Relationship Testing

The *relationship-identification* methodology in ReConnect uses two database techniques—data profiling and schema mapping—to help detect relationships between datasets. Below, we briefly discuss how ReConnect uses these techniques in relationship testing.

Data Profiling

Data profiling is the process of collecting and analyzing statistical summaries of data to understand its structure and content [47]. In the case of database tables, data profiling involves analyzing instances of column values to determine information such as the data type, length, value range, discrete values and their frequency, variance, uniqueness, and occurrence of null values [61]. Data profiling is commonly used in data cleaning, which is the inspection of data sources for the purpose of identifying problematic data [10]. Data-cleaning applications use profiling information to aid in analyzing different aspects of attributes' quality. For example, the min and max values can be used to determine whether the values of a given attribute (column) are within permissible range.

After converting spreadsheets into database tables, ReConnect applies data profiling to analyze instances of column values to determine information such as column data type, length, format, value range, discrete values and their frequencies, variance, uniqueness, and occurrence of null values. (ReDiscover also applies data-profiling to collect such information from the extracted data columns.) The relationship-identification methodology uses this profiling information to aid users in identifying column correspondences between two tabular datasets and as an indicator or counter-indicator for suggesting relationships between datasets.

For example, when dataset A fully corresponds to B , all the corresponding columns in A and B have the same type, and the value ranges of each column in

A are contained in the value ranges of the corresponding column in B , ReConnect suggests the following relationships: 1) A is row-contained in B , and 2) A is a template of B .

Schema Matching

Schema matching is considered an important requirement for applications such as data warehousing, integrating multiple data sources, and XML message mapping [9, 41]. In our work, we are not using schema matching to integrate spreadsheets’ data, nor to transform data from one spreadsheet to another. Instead, we apply the attribute-(column-)correspondence part of schema matching to enable ReConnect to examine columns’ data to confirm or discard suggested relationships.

More specifically, ReConnect relies on SQL queries to examine large datasets in an efficient and scalable manner. But before ReConnect can generate SQL queries, it has to establish a column correspondence between the datasets it is analyzing. ReConnect uses schema matching techniques, namely semi-automated schema correspondence, to produce an initial column correspondence between two tabular datasets, and then relies on the user to correct or confirm the correspondence. ReDiscover also uses schema-matching techniques; however, it applies automatic schema matching that is based on supervised learning techniques, as we discuss in more detail in Section 4.2.4.

2.3.2 Relationship Prediction

The relationship-prediction methodology in ReDiscover uses two machine learning techniques—Conditional Random Fields and Support Vector Machines—to help predict relationships between dataset pairs in a collection of datasets. Additionally, our methodology uses Bloom filters in summarizing and measuring similarity between categorical columns. It also uses Bloom filters to compute features for relationship prediction as we discuss in Section 4.2.5.

Data-Column Extraction

Many scientists use spreadsheets to store and manipulate tabular datasets. However, besides tabular datasets, they often use spreadsheets to store other information, such as pictures, charts, and comments. Thus, in order for ReDiscover to predict relationships among tabular datasets in spreadsheets, it must first identify cells that are part of a data columns and cells that are not.

Conditional Random Fields (CRFs), a framework for building probabilistic models to label sequence or graphical data [49,70], is widely used for tasks of such a nature. CRFs can learn complex, overlapping and non-independent features that operate at multiple levels of granularity. For instance, in the task of table extraction from text documents [59], CRFs use overlapping language and layout features and take neighboring context into account when labeling items (lines of text in that case). CRFs are well matched for our cell-labeling task because spreadsheet cells are rich in layout, language, and context features. These features are important for accurate cell labeling. In Section 5.2, we show that simpler models perform less accurately.

Automated Column Matching and Relationship Prediction

To automate both the process of detecting column matches between the many pairs of datasets it is analyzing, and the process of predicting relationships between these pairs, ReDiscover applies a supervised learning model, namely *support vector machines* (SVMs) [20]. An SVM is a data-driven classification model that applies learning algorithms on training data to recognize patterns. SVMs are widely used in classification tasks, such as handwriting recognition, image classification, and protein classification [12]. We discuss how ReDiscover applies SVM in column matching and relationship prediction in more detail in Sections 4.2.4 and 4.2.5 respectively.

The relationship-prediction methodology can make a better use of a scientist’s time by predicting which pairs of spreadsheets are likely related, and what the relationship between them might be. With the use of Conditional Random Fields in data-column extraction and Support Vector Machines in computing column correspondence and in prediction relationships between pairs of datasets, ReDiscover is fully automated (though the architecture can accommodate human feedback at various points).

Summarizing Categorical Columns

For ReDiscover to match data columns and to predict relationships between dataset pairs without having to extensively analyze their data, it computes statistical summaries for numerical and categorical columns. While using column statistics provides representative features for numerical data, it is not enough to rely on simple count statistics, such as common value frequencies, or counts of unique and null values to summarize categorical data. Some of the statistics do not apply to categorical data (e.g., mean, standard deviation), or may not be very informative (e.g., min, max values). Thus, we developed a technique based on *Bloom filters* that enables us to inexpensively approximate similarity between two categorical columns, as we describe in Section 4.2.4.

A *Bloom filter* is a space-efficient data structure that was originally developed to test for element membership in a set [11]. It is constructed by converting a set of elements $S = \{x_1, \dots, x_n\}$ into a bit vector bf of length m as following. First, the bit vector bf is initialized to zero. Second, k independent hash functions (h_1, \dots, h_k) are defined, each of which maps x_i , for $0 \leq i \leq n - 1$, to one of the bit vector elements (bf_0, \dots, bf_{m-1}). Third, each element x_i of S is fed to each of the k hash functions to get k positions in the vector, which are then set to one.

To test for an element membership, we feed that element to the same k hash functions used to create the bit vector. If all the k bit locations in the bit-vector

are set, then the element “possibly” exists in the set. Otherwise, the element is definitely not in the set. One disadvantage of Bloom filters is that false positives are possible. However, we can tune a Bloom filter to trade the size of its bit vector against its false-positive rate. (See Section 4.2.4 for more details.)

Bloom filters can also be used for collection-to-collection similarity testing. For example, Jain et al. [45] developed a technique that uses Bloom filters to approximate the match between two web documents. First, they used content-defined chunking (CDC) to extract document features. Next, they create a bit vector using these extracted features. Lastly, to determine similarity between two documents, they compare the bit vector of one document with that of the other. Two documents are marked similar if they share a large number of 1’s. In our work, we use Bloom filters to summarize categorical data-columns, and to compute similarity between them, as we explain in detail in Section 4.2.4.

Additionally, Bloom filters can be used for determining containment between collections. If the set bits of the bit vector for collection A are a subset of the bits in the vector for collection B , then there is a high probability that all the elements of A are contained in B . Furthermore, if there is not containment of bits there cannot be containment of collections. In Section 4.2.5, we show how we use this property of Bloom filters to help predict containment between two datasets.

In conclusion, scientists’ activities on their datasets leave behind relationships that are indicative of the original connections between their datasets. Based on this observation, we developed two hypotheses: 1) knowing these relationships can help scientists determine connections between their dataset, and 2) we can at least partially automate the process of relationship identification. To validate these hypothesis, we first developed ReConnect, a tool for testing for relationships between two datasets (Chapter 2.3.2), and evaluated our tool with a user study. The results showed that relationships are useful to researchers and that our methodology can test for relationships efficiently. Encouraged by these results, we expanded our

methodology to deal with a scientist's or a small group of researchers' collections of datasets (Chapter 3.2.3). The ultimate goal of our work is to develop efficient tools for scientists to identify relationships that can help them better understand and work with their data. We discuss the implementation and the evaluation of ReConnect in the next chapter.

CHAPTER 3: RELATIONSHIP TESTING

To validate the utility of our set of relationships in helping scientists with the task of dataset selection, and to see whether or not we can test for relationships in a reasonable amount of time, our initial investigation focused on testing relationships between two datasets. For that purpose, we developed a semi-automated tool, ReConnect, that enables a scientist to test for relationships between pairs of tabular datasets embodied in spreadsheets.

3.1 RECONNECT’S ARCHITECTURE

As shown in Figure 3.1, ReConnect’s architecture consists of two processes. The first process, *Upload Dataset*, converts a dataset in a spreadsheet into a database table (Section 3.1.1). Because a spreadsheet may have multiple datasets, datasets with partial rows or columns, and non-table data, ReConnect allows a user to guide the conversion process, which improves the accuracy with which tabular datasets are extracted. The second process, *Identify Relationships*, involves two tasks: the first is detecting any column correspondence between the two tables (Section 3.1.2), and the second is analyzing the column correspondence and data-profiling statistics to suggest and validate possible relationships between these datasets (Section 3.1.3). Throughout this section, we use the example from Section 2.1.1 to explain how ReConnect works.

3.1.1 Converting Spreadsheets to Database Tables

ReConnect converts a spreadsheet into a database table in two steps. In the first step, *Select Dataset*, a user selects a tabular dataset within her spreadsheet. (It is

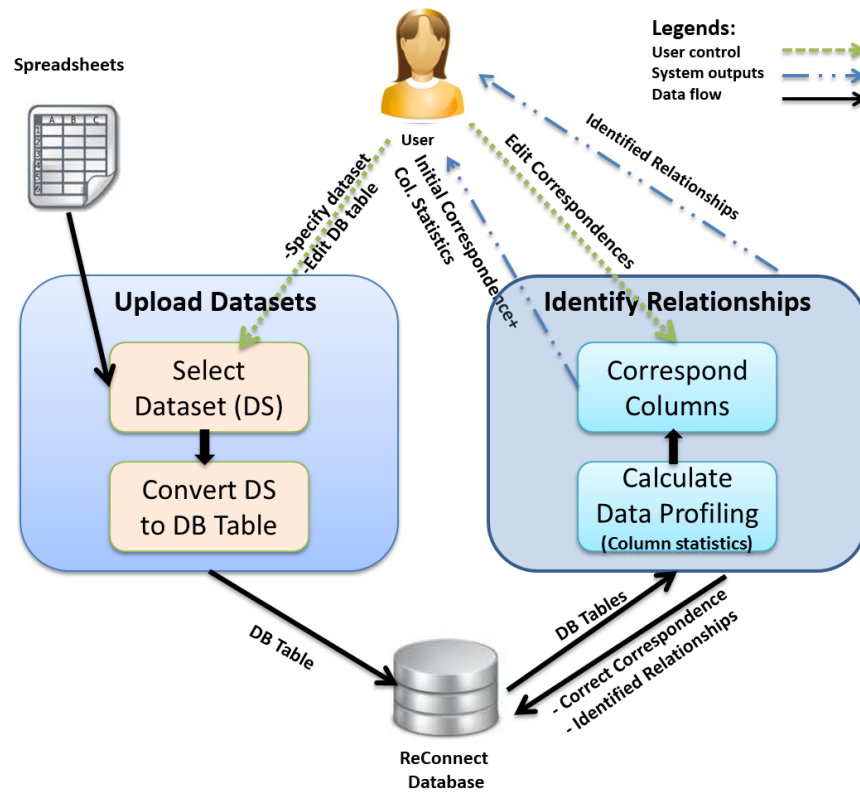


Figure 3.1: ReConnect’s architecture: The process *Upload Dataset* converts a spreadsheet into a database table, and the process *Identify Relationships* detects relationships between pairs of spreadsheets.

also possible for a user to upload several datasets from the same spreadsheet.) As shown in Figure 3.2, Jennifer selects her table by specifying column (1 to 4) and row (2 to 5) ranges. She then selects the index of the row that contains the column names of her table (row 1 contains *B*’s column names). This step is optional, as some tables may not have headers. When she clicks the *Select* button, ReConnect lets her preview the dataset.

In the second step, Jennifer can edit database table information such as the table name and column names. She can also specify a primary key for her table. The tool verifies that a selected column satisfies the constraints on primary keys,

Choose Your File To Upload :

File Name: E:\UploadedUserFiles\B.xlsx
 File Content: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
 File Size: 12105bytes

Enter data range
 Columns: From: To:
 Rows: From: To:

Enter the row index of Column Names

Original Spreadsheet Data					Selected Data				
	1	2	3	4	5	Site	Sample #	Bottle gp	Depth
1	Site	Sample #	Bottle gp	Depth		NH-10	1350	1-3	70.4
2	NH-10	1350	1-3	70.4		NH-10	1351	4-6	60.9
3	NH-10	1351	4-6	60.9		NH-10	1352	7-9	28.7
4	NH-10	1352	7-9	28.7		NH-10	1353	10-12	3.9
5	NH-10	1353	10-12	3.9					
6									
7									by Dr. Jim
8									6:00AM

Figure 3.2: Dataset selection for spreadsheet *B*: Jennifer can specify column and row ranges and the index of the row that contains the column names of her dataset.

such as containing neither nulls nor duplicate values. Once she uploads her data, ReConnect creates a database table and inserts the data in it. ReConnect preserves column order by appending each column's position to its corresponding attribute name in the database table, and preserves row order by adding a new attribute (*row_index*), which maintains the original row order, to the database table.

3.1.2 Column Correspondence

ReConnect must identify correspondences between conceptually identical columns before it can suggest relationships between two spreadsheets. The column correspondence resulting from this process enables ReConnect to suggest relationships

between two datasets, and to compute the set of common rows (in the corresponding columns) between two tables. It also enables the tool to generate SQL queries to examine suggested relationships between the tables.

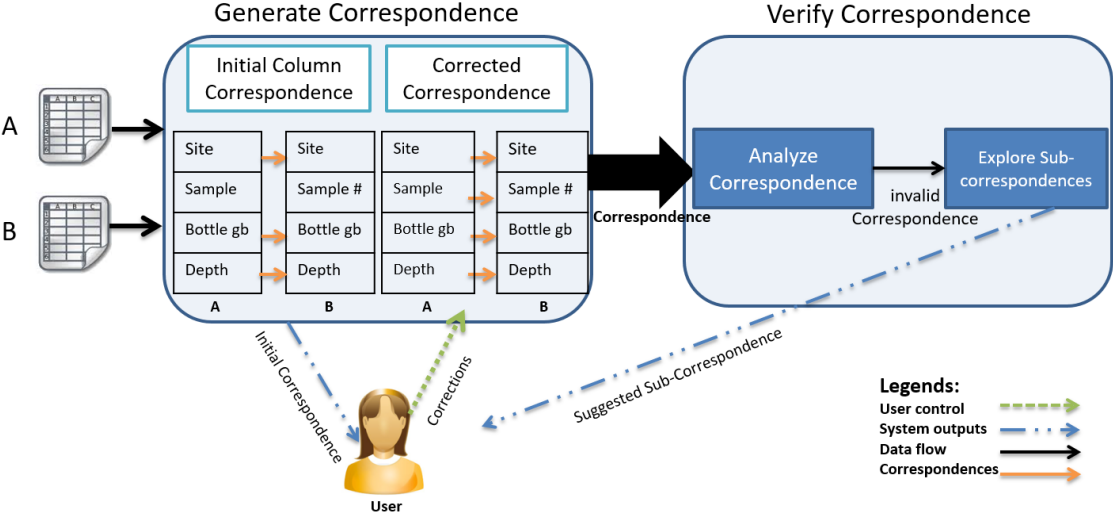


Figure 3.3: *Column Correspondence* process for Spreadsheets *A* and *B*: ReConnect depends on a user’s feedback to identify column correspondences accurately.

Figure 3.3 shows the column-correspondence process for spreadsheets *A* and *B* from our example in Section 2.1.1. ReConnect first generates an initial column correspondence by querying the database for column names from table *A* and *B*, and then matches these names by equality. As shown in Figure 3.4, it outputs the correspondences to Jennifer along with column statistics, including column data type, the number of rows, the number of null values, the number of unique values, the maximum and minimum values, and the common value frequency for each column. These statistics aid her in inspecting and, if necessary, correcting the proposed correspondences. Jennifer notices that columns *Sample* of *A* and *Sample #* of *B* match, so she uses the ReConnect user interface to indicate that these columns correspond. The example shows user feedback enhancing ReConnect’s accuracy in identifying column correspondences.

The screenshot shows the ReConnect web application interface. At the top, there is a navigation bar with 'Upload Dataset' and 'Find Relationships' buttons. Below this is a header section titled 'IDENTIFYING RELATIONSHIPS BETWEEN SPREADSHEETS' with a yellow instruction box: 'Step 2: Please map columns between A and B. To edit column mappings, use 'A Columns' column of 'B Column Statistics' table. Column statistics can aid you in column mapping process.'

The main content area is divided into two sections: 'A Column Statistics' and 'B Column Statistics'. Each section contains a table with the following columns: Column Name, Column Type, Row Count, Count of Unique, Count of Null, Common Value, Common Value Frequency, Minimum Value, and Maximum Value. The 'A Column Statistics' table has four rows: Site (varchar, 2, 1, 0, NH-10, 2, NH-10, NH-10), Sample (float, 2, 2, 0, 1350, 1, 1350, 1351), Bottle gp (varchar, 2, 2, 0, 1-3, 1, 1-3, 4-6), and Depth (float, 2, 2, 0, 60.9, 1, 60.9, 70.4). The 'B Column Statistics' table has four rows: Site (varchar, 4, 1, 0, NH-10, 4, NH-10, NH-10), Sample # (float, 4, 4, 0, 1350, 1, 1350, 1353), Bottle gp (varchar, 4, 4, 0, 1-3, 1, 1-3, 10-12), and Depth (float, 4, 4, 0, 3.9, 1, 3.9, 70.4). There are buttons for 'Clear Column Mappings', 'Analyze Column Mappings', and 'Suggest Relationships'.

Figure 3.4: The initial column correspondence between datasets A and B .

ReConnect uses Jennifer's column correspondences to compute the rows that are common in A and B : the rows with sample numbers 1350 and 1351 (see Figure 2.1). Editing column correspondences may result in different sets of common rows, because ReConnect only checks for row matches relative to the corresponding columns. If there were no common rows in the corresponding columns, ReConnect would suggest that she explore sub-correspondences, where the tool attempts to match a subset of the current column correspondence between A and B looking for a correspondence that produces the largest set of common rows.

For example, suppose that instead of matching $A.Sample \rightarrow B.Sample \#$, Jennifer mistakenly matched $A.Sample \rightarrow B.Site$. ReConnect would find no rows in common for this correspondence. If she chose to use the Explore Sub-Correspondence feature, then ReConnect starts by removing the $A.Site \rightarrow B.Site$ correspondence from the column correspondences and computing the set of common rows by matching row values in the remaining corresponding columns:

$A.Sample \rightarrow B.Site$, $A.Bottle\ gb \rightarrow B.Bottle\ gb$, and $A.Depth \rightarrow B.Depth$. This sub-correspondence will not produce any common rows. Next, ReConnect removes $A.Sample \rightarrow B.Site$ from Jennifer’s column correspondence, and again computes the set of common rows by matching row values in the corresponding columns: $A.Site \rightarrow B.Site$, $A.Bottle\ gb \rightarrow B.Bottle\ gb$, and $A.Depth \rightarrow B.Depth$. This sub-correspondence will produce two common rows: the rows with sample numbers 1350 and 1351. ReConnect continues exploring sub-correspondences until it finds the correspondence that produces the largest number of common rows, which it then suggests to Jennifer.

Relying on column statistics for determining column correspondence between datasets may not be sufficient. There are situations where we can have statistically similar columns that in fact do not correspond. For example, assume spreadsheet X has *Initial Temp* and *Final Temp* columns and spreadsheet Y has *Temp1* and *Temp2* columns. Suppose the *Initial Temp* column closely resemble both *Temp1* and *Temp2* (similarly for *Final Temp*). Simple column-to-column comparison does not give much insight on the correct column-correspondence here. However, comparing row values may help detect which of the two correspondences is appropriate, because row values are more distinctive than single-column values.

3.1.3 Relationship Identification

Figure 3.5 depicts the process of identifying relationships, which involves two steps: *Suggest Relationships* and *Validate Relationships*. In the first step, described in Algorithm 1, ReConnect attempts to suggest relationships by analyzing column correspondences and column statistics to check if they are compatible with the conditions for a relationship.

The result of this step is a “quick and dirty” list of possible relationships—quick because the analysis does not involve the individual data values in the datasets, and dirty because statistics and column correspondences may not be sufficient to

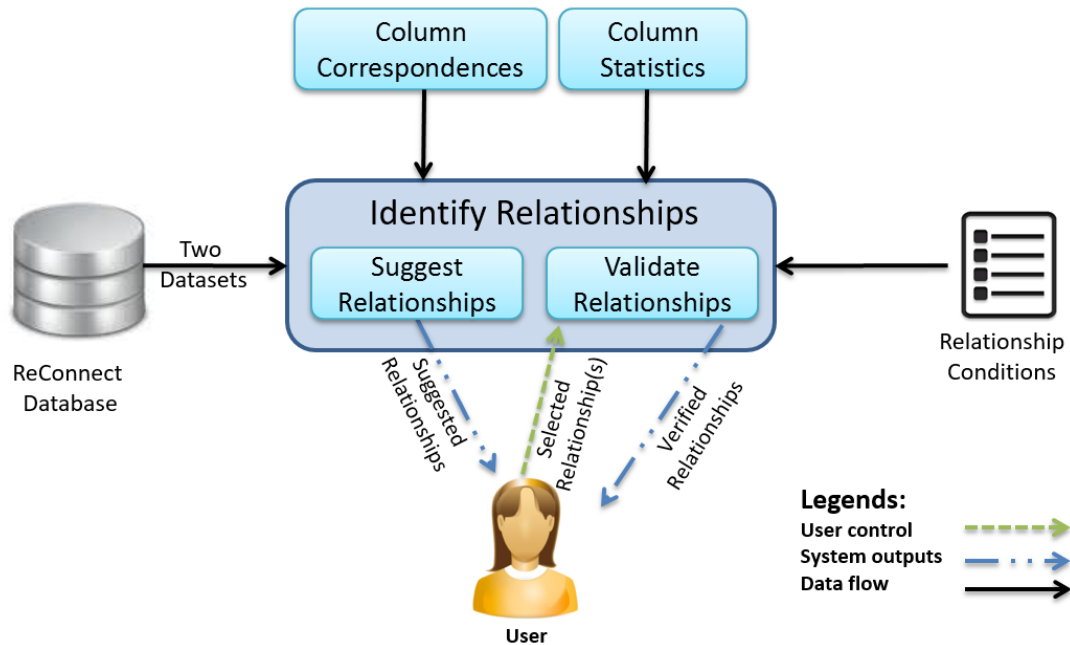


Figure 3.5: The *Identify Relationships* Process: ReConnect analyzes column correspondences, column statistics, and the set of common rows of two tables to check if they match the conditions for a relationship

validate certain relationships. For instance, if the number of rows of one spreadsheet is greater than the number of rows of another, then it is possible that the relationship between these spreadsheets is *Row-containment*. However, for ReConnect to validate such a relationship, it must check whether or not all of the data rows of the first spreadsheet are contained in the second spreadsheet. The purpose of the list of suggestions is to provide users with hints about possible relationships without having to analyze the actual data. It also removes from consideration relationships that cannot possibly hold under the current column correspondence and statistics.

Figure 3.6 shows how the *Relationship Identification* methodology classifies

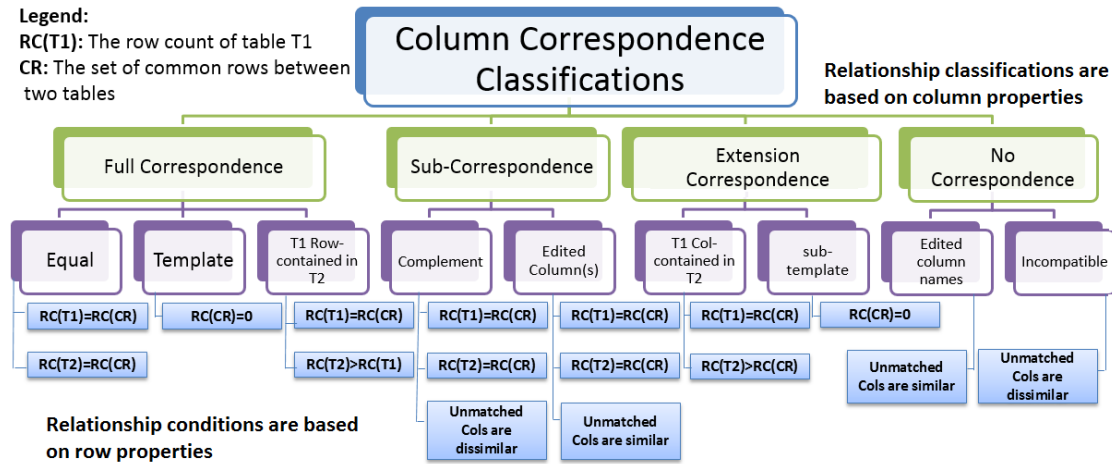


Figure 3.6: Relationship Identification Overview: Classifying relationships based on column correspondences helps ReConnect limit the relationships to investigate.

relationships based on properties of a column correspondence. Classifying relationships based on column correspondences, such as *Full Correspondence*, *Sub-correspondence*, and *No Correspondence*, helps ReConnect limit the number of relationships it investigates for a pair of datasets. The figure additionally shows that relationship conditions are also based on row properties such as row count and row-value similarity.

During the second step, *Validate Relationships*, the user can choose a relationship to investigate from the list of suggested relationships. ReConnect validates the selected relationships by generating an SQL query that examines the two tables and then analyzing the results of the query, in addition to column correspondences and statistics. The details of the *Validate Relationships* process are presented in Algorithm 2.

For example, since there is a full correspondence between the columns of Jennifer’s spreadsheets *A* and *B*, in the *Suggest Relationships* step, ReConnect analyzes column statistics to determine which of the three relationships applies: *Equal*, *Template* or *A is Row-contained in B*. The tool obtains the row count of *A* and *B*

Algorithm 1 Suggest Relationships algorithm

Input: Column correspondences (col_corr); column statistics (col_stats) and the set of common rows statistics (CR_stats).

Output: List of suggested relationships ($suggested_relts$).

1. $col_corr_class = classify_correspondence(Col_Corr)$; ▷
 $classify_correspondence()$ takes the column correspondences and returns Full Correspondence, Extension Correspondence, Sub-correspondence or No Correspondence
 2. $possible_relts =$ get the relationships associated with the col_corr_class classification;
 3. **for** each relationship in $possible_relts$ **do**
 4. $relt_conds =$ get the relationship conditions;
 5. **if** col_stats and CR_stats are compatible with $relt_conds$ **then**
 6. $suggested_relts.add(relationships)$;
 7. **end if**
 8. **end for**
-

($RC(A) = 2$ and $RC(B) = 4$). Since the number of rows between A and B is different, the *Equal* relationship cannot hold. Because $RC(B) > RC(A)$ ReConnect adds *A is row-contained in B* to the list of suggested relationships. It also adds the *Template* relationship based on the column correspondence.

Suppose that Jennifer selects the *A is row-contained in B* relationship for validation. The *Validate Relationships* algorithm starts the validation process by looking up the validation tests associated with the selected relationship. As part of each validation test, ReConnect generates an SQL query that examines certain relationship features. For instance, to count the common rows between A and B ,

Algorithm 2 Validate Relationships algorithm

Input: User-selected relationship (*suggested_relt*); relationship conditions (*relt_conds*); column correspondence (*col_corr*) and tables *A* and *B*.

Output: Confirmation or invalidation of the relationship.

1. *valid_test* = get the validation tests for the selected relationship *suggested_relt*,
 2. **for** for each test in *valid_tests* **do**
 3. *test_qry* = *generate_qry(col_corr, suggested_relts)* ▷
 Based on predefined validation test, *generate_qry()* uses the column correspondence to generate an SQL query that is used to test the tables data for the suggested relationship.
 4. *test_results* = *run_query_in_DB(test_query)*;
 5. *results_stats* = *analyze_results(test_results)*; ▷ *analyze_results()*
 collects statistics and other information about the results of the test query, which ReConnect uses to validate the suggested relationship.
 6. **end for**
 7. **if** all validation tests passed **then**
 8. Confirm relationship;
 9. **else**
 10. Invalidate relationship;
 11. **end if**
-

ReConnect generates the following query (which handles repeated rows by grouping and counting them in each dataset). The query joins tables *A* and *B* based on the column correspondence, and then counts the set of common rows between the two tables ($RC(CR) = 2$). Since $RC(B) > RC(A)$ and $RC(A) = RC(CR)$,

Reconnect reports to Jennifer that the row_containment relationship between A and B is confirmed.

```

SELECT SUM(MIN2(t1.Cnt,t2.Cnt)) AS RC_CR
FROM (SELECT ta.Site, ta.Sample, ta.[Bottle gp], ta.Depth, COUNT(*)
      as Cnt
      FROM A ta
      GROUP BY ta.Site, ta.Sample, ta.[Bottle gp], ta.Depth) t1,
      (SELECT tb.Site, tb.[Sample #], tb.[Bottle gp], tb.Depth, COUNT(*)
      as Cnt
      FROM B tb
      GROUP BY tb.Site, tb.[Sample #], tb.[Bottle gp], tb.Depth) t2
WHERE t1.Site=t2.Site and t1.Sample=t2.[Sample #] and t1.[Bottle
      gp]=t2.[Bottle gp] and t1.Depth=t2.Depth
--MIN2(Arg1,Arg2) returns the minimum value out of the two arguments
      passed to it.

```

The Implementation of ReConnect

We initially implemented ReConnect as a web application running on a local Microsoft .Net server. The user interface (UI) of ReConnet, including Upload Datasets and Identify Relationships, was developed in C#, and the database was implemented in Microsoft SQL Server 2008. To integrate some of ReConnect functionality with SQLShare system, we later converted it into a Windows Azure cloud application, and migrated its database to Azure SQL Server platform.

As we discussed above, once the user selects a dataset, ReConnect converts it to DB table. Then, ReConnect uses a set of static stored procedures to compute data-profiling information for the uploaded dataset. These stored procedures run in ReConnect's Microsoft SQL Server database. Then, when a user selects

two datasets to test for relationships, ReConnect computes the initial column correspondence, and allows users to modify the correspondence through the UI, as shown in Figure 3.4. Based on the column correspondences and column statistics, ReConnect suggests possible relationships to the user, who then selects which relationship to validate. Depending on the selected relationship, ReConnect generates several dynamic SQL queries, which are executed in ReConnect’s database. Lastly, it analyzes queries results to confirm or invalidate the selected relationship.

In the next section, we discuss the results of our preliminary evaluation of ReConnect, and discuss our user study, which provided valuable feedback from scientists about ReConnect and the set of relationships it identifies.

3.2 USER STUDY AND EVALUATION

The assement of our initial work has two parts. First, we conducted a user study to asses ReConnect and the set of relationships it identifies. The goal of this study was to get initial answers to the following research questions.

- RQ1 Do the relationships ReConnect detects actually turn up in real-life spreadsheets?
- RQ2 When subjects use spreadsheets, do they confront problems that ReConnect seeks to solve?
- RQ3 Does detecting relationships between spreadsheets help subjects select and manipulate their data?
- RQ4 Are there other kinds of relationships we should consider detecting?
- RQ5 Do aspects of spreadsheets that ReConnect cannot currently handle interfere with detecting relationships?

Second, we compared the effectiveness of ReConnect and four commercial and research change-inference tools for spreadsheets in identifying relationships between spreadsheets.

3.2.1 User Study¹

In our study, we asked researchers to attempt to identify relationships between their spreadsheets with and without the help of ReConnect. The purpose of this study was to assess the applicability of the concept of relationship identification, the usefulness of ReConnect in simplifying the task of detecting relationships between spreadsheets, and to know whether our relationships arise in real-world scientific spreadsheets. Our user study consisted of three parts. In the first part, we asked subjects to manually detect relationships between their spreadsheets. In the second part, subjects used ReConnect to do the same task they had performed in part one using the same datasets. In the final part, we conducted interviews with subjects about their experience with the relationship-identification task, both with and without the use of ReConnect.

Subjects To recruit subjects for the study, we sent emails to mailing lists for faculty and graduate students in several science departments at different universities and research labs. We recruited 10 subjects from various science fields such as Biology, Chemistry, Economics, Agriculture, and Computer Science. Macefiled [54] indicates that a size of 10 is effective for evaluating early conceptual prototypes. He states that “In the case of studies related to problem discovery in early conceptual prototypes, there are typically factors that drive the optimal group size towards the lower end of this [3-20] range” [54].

Prior to each user session, we asked the subjects to send us a pair of spreadsheets they use in their research work. We wanted to make sure that their spreadsheets

¹We had Portland State IRB approval for our user study.

contained tabular datasets. We also asked for spreadsheets that the user believed were connected, though we did not require that they be certain of the existence or the exact nature of the connection. Thus, we were able to evaluate ReConnect using real-life research spreadsheets that subjects provided.

Methodology We conducted the study using a desktop computer with dual wide-screen monitors, which enabled subjects to view their spreadsheets side-by-side to facilitate the manual inspection of relationships. The computer ran Microsoft Excel 2010 on Windows 7 Enterprise Edition. Sessions were conducted on a one-on-one basis, where I supervised the session. We describe the three parts of a session in more details below.

Part One: Detecting Relationships Manually

For each subject, we first explained the concept of *relationship identification* using the example spreadsheets shown in Figure 2.1. We also introduced the set of relationships that ReConnect identifies. Then, each subject was asked to visually inspect the spreadsheets he or she provided, looking for relationships that could help him or her understand how the spreadsheets connect to each other. Throughout this part of the study, we took notes on how the subject inspected his or her spreadsheets, and answered the subject's questions.

Part Two: Detecting Relationships Using ReConnect

We first demonstrated ReConnect using spreadsheets shown in Figure 2.1. Next, subjects used ReConnect to convert their spreadsheets into database tables, and then to detect relationships between their spreadsheets. During this session, we provided minimal support for the subjects, and took notes on how they interacted with the tool. Using ReConnect, subjects identified several relationships, including

reordered rows and columns, equal, row-containment, column-containment, complementation, and template relationships between their datasets.

Part Three: Interviews

In the last part of each session, we interviewed each subject to obtain his or her feedback about the task of identifying relationships with and without the use of ReConnect. The main goal of this interview was to find out how helpful detecting relationships was in deciding how to work with their spreadsheets, to assess the helpfulness of ReConnect in detecting relationships, and to determine whether there were other relationships of interest that ReConnect did not handle. Additionally, to determine whether our relationships arise between scientific datasets, we first introduced these relationships to our subjects using several synthetic dataset pairs. Then, we asked each subject whether he or she believed that these relationships could exist between scientific datasets that they have encountered in their research.

Results Even with the aid of wide-screen dual monitors and the vertical and horizontal side-by-side view feature that MS-Excel provides, all subjects found it difficult to visually inspect spreadsheets for relationships. Most difficulties arose from inspecting spreadsheets with a large number of columns and rows or differently ordered rows and columns, and from attempting to detect whether or not two spreadsheets share a subset of their rows or columns (or both). As a result, subjects sometimes reported relationships incorrectly between their spreadsheets. For example, a chemistry researcher thought that she had edited a number of rows in one of her spreadsheets and saved it with a different name. In fact, the rows of the two spreadsheets were identical but appeared in a different order. However, she was able to correctly identify that she had added columns that represented new data for samples.

In a few cases, subjects were able to “guess” such relationships as column-containment and row-containment. However, they were not able to confidently confirm the results of their observations. For instance, while visually inspecting her spreadsheets, a computer scientist remarked, “Based on the number of rows in both spreadsheets, I guess the relationship is row-containment. But to confirm that, I need to write a VBA (Visual Basic for Applications) script.” However, writing VBA scripts might be a difficult task for users with no programming experience. Overall, subjects were unable to confidently identify relationships between their spreadsheets through visual examination.

With ReConnect, subjects quickly and effectively detected useful relationships, which enabled them to recall the activities they performed to transform one version of a spreadsheet into another, and to decide how to further reuse or combine their datasets. For the physics researcher, in addition to confirming the column containment relationship she detected manually in part one, she found that the rows of the spreadsheet with more columns had been filtered and reordered (to facilitate the analysis of an experiment she was conducting).

Besides detecting relationships, the physics and chemistry researchers found that some of ReConnect’s features could be used for data analysis. For instance, after analyzing several correspondences between the columns of her spreadsheets, a chemistry researcher stated “Often, I repeat experiments with minor changes in my experiment configuration, such as increasing the temperature of the sample environment by 10 degrees. Using this tool, I can test several column correspondences to analyze the effect of such configuration changes on various experimental results.”

Discussion The third part of the session (interviews) suggests that the relationships that ReConnect detects do actually exist between scientific datasets (RQ1). Subjects described several activities they regularly perform on spreadsheets that

can produce the types of relationships that ReConnect detects. For example, one subject stated that she often filters a dataset based on several criteria, and then saves each version as a new spreadsheet. Such activities produce the row-contaminant relationship between the original dataset and all the derived (filtered) versions. Another subject stated that he usually performs several experiments on the same set of chemical samples, and then stores each experiment on a separate dataset. Such actions give rise to the complementation relationship, as this set of datasets provide complementary information about the same set of samples.

Most subjects had affirmative responses regarding whether, when working with spreadsheets, they face problems that ReConnect can help solve (RQ2). One subject stated that she does not often face such problems because her spreadsheets contain reference information that is rarely updated. However, she stated that she previously confronted similar problems when she worked on a collaborative research project.

The first two parts of the sessions suggest that ReConnect significantly simplified the task of identifying relationships between spreadsheets, a positive answer to RQ3. All subjects agreed that determining relationships removed the burden of comparing and analyzing spreadsheet cells, columns, and rows. Results also suggest that subjects found value in using ReConnect to aid in data-analysis tasks, which is a potential direction for future work.

Subjects suggested a number of relationships to add to ReConnect's identification capabilities (RQ4). For instance, one subject suggested detecting pairwise column equality within a given percentage range (*near-match* relationship), which could aid her in analyzing her spreadsheets' data. For instance, she might want to detect all the rows whose temperature columns are equal within $\pm 2\%$. Another subject suggested identifying datasets with calculated columns that contains formulas with reference to other datasets. This relationship is very similar to the *cell/sheet reference* feature provide by Microsoft Inquire.

Regarding whether some aspects of spreadsheets might interfere with detecting relationships (RQ5), a single subject stated that cell formulas could interfere with detecting relationships, as they may present irrelevant information about the original spreadsheets’ data.

While the limited number of subjects may not enable us to detect all possible issues with ReConnect, the group size was sufficient for early problem discovery. The importance of this study was that it verified that there are no major issues with the tool, nor with our assumptions about the usefulness of the relationship-identification methodology. It is also worth mentioning that ReConnect response time was not an issue, for the size of spreadsheets our users had. The SQL query formulation and evaluation generally gave response times less than 10 seconds.

To sum up, we had positive feedback from our user study participants about the usefulness of ReConnect and the set of relationships it identifies. The results of our study suggest that subjects perform activities on their spreadsheets that give rise to most of our relationships. It also confirmed that scientists often struggle with identifying which dataset(s) to select, or how to work with the data stored in their datasets, and that detecting relationships can help with these tasks. Additionally, subjects proposed detecting new kinds of relationships that could help with the selection task, including the near-match and *cell/sheet reference* relationships. Regarding whether any of the spreadsheet aspects could interfere with detecting relationships, one subject suspected that spreadsheet formulas may complicate the process of relationship-identification. We plan to investigate this possibility in future work.

3.2.2 Methodological Evaluation

Since there are no commercial or research tools that are aimed specifically at discovering relationships between spreadsheets (or tabular datasets more generally), we looked at other tools that may help scientists with this task. Change-inference

tools, which enable users to identify changes between pairs of spreadsheets, much like “diff” utilities on documents, have some capabilities that might help with the task of discovering connections between datasets. The aim of this evaluation was to investigate whether there were any existing commercial or research tools that could provide the same capabilities as ReConnect.

Selected tools We selected one research change-inference tool (SheetDiff [16]) and three commercial tools (DiffEngineX [34], Synkronizer [75], and Excel Inquire [56]) for our evaluation. These tools generate a report of differences and either highlight differences between spreadsheets in both spreadsheets, as is the case with Synkronizer, DiffEngineX, and Inquire, or do so only in one spreadsheet, as is the case with SheetDiff.

Methodology We used two sets of spreadsheets for our investigation. The first set was a collection of related pairs of real-life research spreadsheets that our user-study participants had provided. Since the relationships between each of these pairs had been identified and confirmed during the user study, we used this set to evaluate whether or not change-inference tools would help us get to the same results we had obtained in the user study.

The second set contains 10 pairs of spreadsheets that we selected from the EUSES corpus [33] and from other sources. This set was constructed to test combinations of relationships and spreadsheet features that are not covered by the first set. For instance, if we tested the containment relationship only between small spreadsheet instances from the first set, then we selected pairs of spreadsheets with large dataset instances that have the containment relationship between them for the second set. We also modified some of the selected pairs to test for special relationships cases such as *infix*, *prefix*, *suffix*, *reordered rows*, *reordered columns*, and *reordered rows and columns* relationships.

We started by grouping spreadsheet pairs based on the relationship they represent. Next, we used these pairs as the input to each of the selected tools. Each tool was tested with small and large spreadsheet instances. Then, we analyzed the results of each tool to evaluate how readily these results can be used to confirm the existence of the previously identified relationship between the inputted pairs.

3.2.3 Results

The results of our experiment are summarized in Table 3.1. Regarding the *duplicate* relationship, all tools were able to detect it for both large and small spreadsheet instances. On the other hand, only ReConnect were able to detect *reordered rows* for both large and small spreadsheet instances. The user can easily identify duplicate pairs when change-inference tools reported no changes. For the *cell reference* relationship, only Inquire has the capability to provide a graphical representation (relationship diagram) of links (formula references) between the current workbook (worksheet, cell) and all other workbooks (worksheets, cells). Such a feature might help users determine connections between business spreadsheets, such as budgets and financial statements. However, in the case of scientific spreadsheets, cross-dataset references are rarely used.

Regarding the containment relationships, including *row containment*, *column containment*, and *sub-containment*, only ReConnect was able to identify them between both small and large spreadsheet instances. All other tools were able to help in detecting containment relationships only between small instances. For example, users may identify a *column-containment* relationship by analyzing the additional columns that Synkronizer reported. However, for large spreadsheet instances, change-inference tools generate a large list of changes that are hard to comprehend, and hence complicate the relationship-identification task. For example, DiffEngineX reported 1472 changes between an 80-row spreadsheet and its

Table 3.1: The methodological evaluation results

Relationship	DiffEngineX	Synkronizer	SheetDiff	Inquire	ReConnect
<i>Duplicate</i>	+	+	+	+	+
<i>Cell/Sheet Reference</i>	–	–	–	+	–
<i>Row Containment</i>	√	√	√	√	+
<i>Column Containment</i>	√	√	√	√	+
<i>Sub-Containment</i>	√	√	√	√	+
<i>Augmentation</i>	√	–	√	√	+
<i>Complementation</i>	–	√	√	√	+
<i>Template</i>	√	–	√	√	+
<i>Infix/Prefix/Suffix</i>	–	√	–	–	+
<i>Subsequence</i>	–	√	–	–	+
<i>Reordered Rows</i>	–	√*	–	–	+
<i>Reordered Columns</i>	–	√*	√*	–	+
<i>Reordered Rows/Cols</i>	–	–	–	–	+
<i>Incompatible</i>	√	√	√	√	+

⁺ The tool can identify relationships in both small and large instances of spreadsheet pairs.

√ the tool can only identify relationships in small instances of spreadsheet pairs.

– The tool helps identify relationships in neither small nor large spreadsheet pairs.

* Results are presented in terms of deleted and added rows and columns. For small spreadsheets instances, users may be able to infer that two spreadsheets are actually the same with different row and column order.

row-extended and reordered version. ReConnect would report as a single relationship.

Spreadsheet B				Spreadsheet B' (Reorderd version of B)			
Site	Sample #	Bottle gp	Depth	Site	Sample #	Depth	Bottle gp
NH-10	1350	1-3	70.4	NH-10	1350	70.4	1-3
NH-10	1351	4-6	60.9	NH-10	1353	3.9	10-12
NH-10	1352	7-9	28.7	NH-10	1352	28.7	7-9
NH-10	1353	10-12	3.9	NH-10	1351	60.9	4-6

Figure 3.7: An example of two equal spreadsheets with reordered rows and columns: Change-inference tools report that spreadsheets B and B' are not equal (identical). ReConnect reports that they are *equal* regardless of their row and column order.

None of the change-inference tools were able to help in detecting complex relationships, such as *reordered rows* or *reordered columns*, where two spreadsheets contain identical but rearranged data. Spreadsheet B and its reordered version, B' , shown in Figure 3.7, illustrate the issue of order sensitivity. As shown in Figure 3.8, SheetDiff reported that columns *Sample #* and *Bottle gp* each have two unmatched cells (the second and the fourth cell). SheetDiff also reported that column *Depth* of B' has been added to B and that *Depth* of B was deleted without noticing that the deleted and added columns are actually the same with different row order.

Similarly, as shown in Figure 3.9, Microsoft Inquire exhibited the order sensitivity issue as it reported that row 5 was added to A , and there are six cells with “Entered Value Changed”. However, since ReConnect uses schema correspondence, it matched columns *Depth* and *Bottle gp* of spreadsheet B with their counterparts in spreadsheet B' in spite of their different orders. ReConnect used this correspondence to compute the set of common rows without sensitivity to the rows’ order. Because the columns of spreadsheets B and B' fully correspond, and

	A	B	C	D	E
1	Site	Sample #	Depth	Bottle gp	Depth
2	NH-10	1350	70.4	1-3	70.4
3	NH-10	1351	3.9	4-6	60.9
4	NH-10	1352	28.7	7-9	28.7
5	NH-10	1353	60.9	10-12	3.9

Figure 3.8: SheetDiff results of comparing spreadsheets B and B' (shown in Figure 3.7). Yellow is used for changed cells, blue for added columns and rows, and red for deleted columns and rows.

the row counts of B and B' are equal to the row count of the set of common rows, ReConnect reported that spreadsheets B and B' are *equal*.

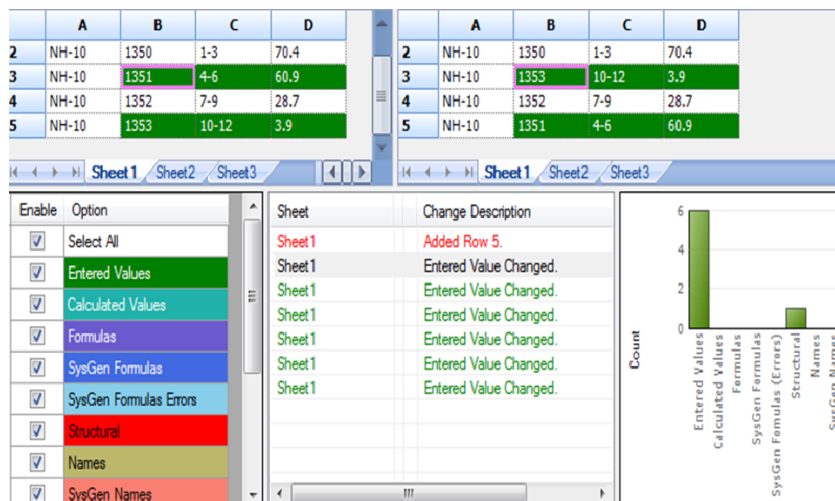


Figure 3.9: Microsoft Inquire results of comparing spreadsheets B and B' (shown in Figure 3.7).

Even with small spreadsheet instances, most change-inference tools reports were not useful in detecting order-sensitive relationships, such as *infix*, *prefix*, *suffix*, *subsequence*, and *reordered columns and rows*. Only Synkronizer and SheetDiff were able to help in detecting some of these relationships in small spreadsheet

instances, due to their ability to aggregate cell changes into higher level changes (e.g., added rows, deleted columns). However, Synkronizer was able to detect four out of the five order-sensitive relationships, where SheetDiff was able to detect only one. Synkronizer’s advantage is its ability to identify when row (or column) positions have changed between two spreadsheets. However, Synkronizer failed to identify equal spreadsheets when both rows and columns were reordered.

While change-inference tools do not require user interaction as ReConnect does, it is still difficult for users to use these tools in the task of understanding relationships between their spreadsheets, because the larger a pair of spreadsheet instances is, the longer the list of changes users have to analyze. ReConnect offers a “set at a time” approach to disambiguating a sea of spreadsheets, while the other tools seem to be row-, column- or cell-oriented — which does not scale as spreadsheets become larger. In addition, the user-verified schema correspondence allows ReConnect to easily identify order-sensitive relationships. ReConnect is working on a more conceptual level than other change-inference tools, and hence reports changes in a more abstract and compact form.

In conclusion, the work on ReConnect confirmed the following. First, relationships do exist in real-life scientific datasets, and detecting them can help scientists determine connections between their datasets. Second, we can efficiently semi-automate the relationship-identification process. While it may not be feasible for scientists to use ReConnect for determining relationships between datasets in a collection, it remains a useful tool for investigating a pair of datasets. In the next chapter, we discuss our relationship-prediction methodology that helps scientists identify the pairs that are mostly likely related, and predicting the relationship between them. ReConnect can be used to confirm or invalidate these predicted relationships.

CHAPTER 4: RELATIONSHIP PREDICTION

In our initial work, where we considered just pairs of spreadsheets, we were able to validate that the idea of relationship identification can help scientists with the task of selecting which datasets to work with, and that by using database technology we can test for relationships in a reasonable amount of time. But looking at real-life research settings, scientists generally work with more than two datasets—often with collections of dozens of spreadsheets.

Our first tool relied to some degree on the user’s help with the table-extraction and column-matching processes. So, even though ReConnect partially automated the process of relationship identification, which is tedious and error-prone when attempted manually, it is still time consuming for a scientist to use ReConnect to compare all possible pairs in a large collection of spreadsheets, to figure out which spreadsheets are related.

Thus, we sought a more automated approach to handle collections of datasets, and we decided that by predicting which pairs of spreadsheets were likely related, and what the relationship between them might be, we can make a better use of scientist’s time [6]. Now our research question becomes: given a collection of spreadsheets, how far we can go in predicting relationships between pairs without user involvement? We now discuss some the challenges we faced in answering this question.

4.1 CHALLENGES

Upgrading our interactive approach to a fully automated one raised a number of challenges, such as extracting datasets from spreadsheets, scaling to many datasets,

and matching data columns. In this section, we discuss the details of these challenges, while we discuss our solutions to the challenges in the following sections.

4.1.1 Automating the Extraction of Tabular Datasets from Spreadsheets

Before the relationship-identification methodology can predict relationships in large collections of datasets, it must first extract tabular datasets without user involvement. However, because of the many variations in dataset layouts and the mixing of non-text data with tabular datasets, extracting datasets from spreadsheets is non-trivial. Additionally, using machine-learning techniques, such as Conditional Random Fields (CRFs), to automate the extraction of datasets can entail some technical difficulties, as we discuss below.

While the better the accuracy of the dataset-extraction stage, the better the quality of the results of later stages (such as column matching and relationship prediction), dataset extraction does not have to be faultless. For instance, even if our approach misses a row when extracting a dataset, it might still be able to make reasonable predictions.

Wide Variety of Dataset Layouts in Spreadsheets

Spreadsheets give users freedom of expressiveness in storing and manipulating their datasets. However, such flexibility often results in a wide variety of layouts for datasets. For instance, a dataset may have spanning headers, columns separated by gaps (empty cells), or multiple tables per sheet. As a result of such ad-hoc data layouts in spreadsheets, dataset-extraction methods must cope with a wide variety of such layouts.

Non-data Text

Spreadsheets often hold other information besides datasets, such as charts, notes and comments, which are often combined with data tables. Our relationship-prediction methodology must be able to exclude such non-data elements based on more than just cell contents. For instance, a cell may contain a text string that could be a string value in a column, a column header, or a non-data comment or note. Thus, we need a method that can incorporate cell context as well as cell content. Later in this chapter we will see that CRFs are able to take advantage of such information in distinguishing between data and non-data cells.

Producing Training Data for Cell Labeling

While we can automate the task of dataset-extraction by using machine learning methods, such as CRFs and Hidden Markov Models (HMM), for any such methods to cope with the variety of layouts and non-data elements in spreadsheets, it will need a large training set. For instance, in our cell-labeling task (Section 4.2.1) we need to label cell types as part of building the training set, which necessarily requires human judgment. Consequently, this task can take several hours for even a small dataset. Further, errors in cell labeling are very likely, due to the intensive manual work.

4.1.2 Scaling to Many Datasets and Columns

Scientists can accumulate collections of dozens or even hundreds of datasets. Processing such datasets for relationship prediction can be time-consuming. To work efficiently with large collections of datasets, our relationship-prediction methodology must scale to many datasets and columns. To achieve this goal, our methodology should favor per-dataset and per-column computations and minimize computation in joint pairwise tests. We discuss the scalable architecture of our approach

below.

4.1.3 The Need for Sophisticated Column Matching

For our methodology to match data columns and to predict relationships between datasets without having to extensively compare individual data values, it computes statistical summaries for numerical and categorical columns. From our experience with ReConnet, we know that most of the statistics we had for summarizing numerical columns did not apply for categorical data, and the one that did were not very informative. Furthermore, before computing column summaries, our methodology has to infer the type of each data column, which is a challenging task. We discuss these challenges below.

Inferring the Type of a Column

Inferring the type of a data column in a spreadsheet can be difficult for the following reasons. First, while spreadsheet data-cells can be formatted as date, number, text, currency, etc., cell formatting is not always consistent in a column. Second, spreadsheet users include notes and comments along with data values to describe anomalies, to make interesting observations or to indicate the absence of certain values. For instance, a researcher may use several terms, such as *NA*, *Unknown* and *No Reading*, to indicate the absence of a temperature reading for a given sample. As a result, data columns may have values of inconsistent types.

Summarizing Categorical Columns

Categorical columns have string values from a limited (often fixed) domain, such as animal species, blood types, and states or regions. Numerical columns are integer or float data for observations that can be measured, e.g., body temperature. While column statistics provide representative summaries for numerical data, the statistics available for categorical data, such as common value frequencies, counts

of unique and null values, and min and max values, do not provide much insight in our experience. Furthermore, statistics such as mean and standard deviation are not even sensible for or compatible with categorical data. Additionally, because our methodology has to compare a large number of columns, methods for comparing categorical data that extensively compare individual column values, such as edit distance and Dice coefficient, are expensive, and hence we preferred to avoid them. Thus, in order to compare categorical columns efficiently, we would like to compute bounded-size column summaries inexpensively that provide a representative approximation of the values in these columns.

4.2 A DESCRIPTION OF THE REDISCOVER SYSTEM

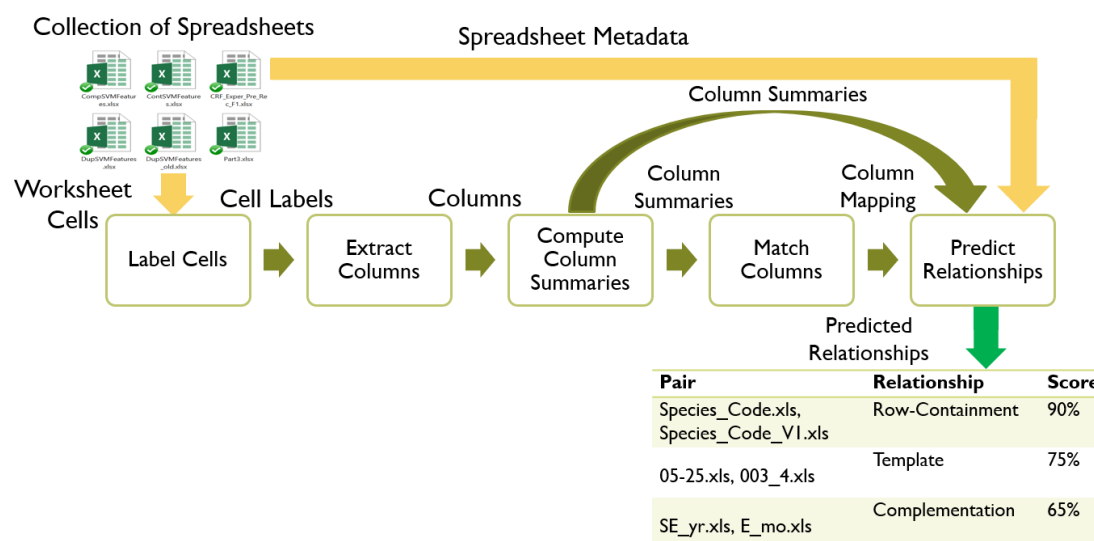


Figure 4.1: The main processes in ReDiscover

We developed ReDiscover, a prototype system for predicting relationships in a large collection of spreadsheets, to help explore a more automated approach. A scientist can use ReDiscover to identify the pairs that are most likely related, and then he or she can use ReConnect to confirm ReDiscover’s predicted relationships. We provide an overview of ReDiscover in this section. We discuss ReDiscover’s

experimental evaluation in Chapter 4.2.5.

In Figure 4.1, we show the component architecture of ReDiscover, which consists of five processes: *Label Cells*, *Extract Columns*, *Compute Column Summaries*, *Match Columns*, and *Predict Relationships*. First, Label Cells (Section 4.2.1) assigns a label to each spreadsheet cell that indicates whether or not it belongs to a data column. Second, Extract Columns (Section 4.2.2) uses cell labels to find groups of vertical cells that likely constitute data columns. Third, Compute Column Summaries (Section 4.2.3) computes statistical summaries for the extracted numerical and categorical columns. Fourth, Match Columns (Section 4.2.4) uses these summaries to identify possible column correspondences for each pair of datasets. Finally, ReDiscover predicts relationships based on column summaries, spreadsheet metadata, and column correspondence (Section 4.2.5).

We developed ReDiscover with a scalable architecture to work efficiently with large collection of datasets. Label Cells, Extract Columns, and Compute Columns Summaries operate on a per-dataset basis. Match Columns and Predict Relationships are the only processes where joint pairwise dataset features are computed. Thus, for example, Compute Column Summaries can derive summaries of multiple columns in parallel. The processing in the first three steps is proportional to the table size of the spreadsheets and can be parallelized easily. Consequently, we tried to minimize the amount of work that ReDiscover has to do at the end for the n^2 pairs of datasets.

4.2.1 Label Cells

As we discussed in Section 2.3.2, extracting data columns from spreadsheets can be challenging, as there is often non-data information present, such as comments, and the datasets themselves vary widely in format. Hence, the goal of Label Cells is to identify cells that are part of a data column and label them accordingly.

Figure 4.2 depicts both Label Cells’ training and application phases. The

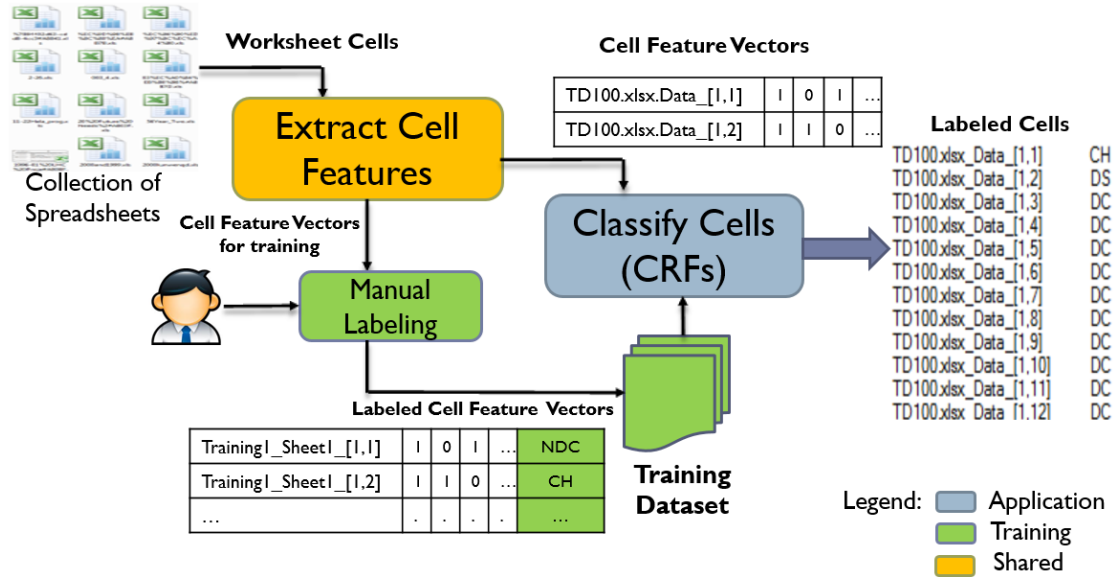


Figure 4.2: *The Label Cells Process*

Extract Cell Features process, used in both phases, extracts for each worksheet cell a k -dimensional *cell feature vector* (currently $k = 40$) that describes the cell’s layout, text, content, and context information. In the application phase, using the Linear Chain version of CRFs, ReDiscover then scans cells vertically² and the *Classify Cells* process assigns one of the following labels to each cell:

CH (Column Header): cells that contain column headers.

DS (Data Start): the first data cell in the column.

DC (Data Continuation): any data cell between the start and the end cell.

DE (Data End): the last data cell in the column.

NDC (Not a Data Cell): a cell that does not belong to a data column.

We defined these labels to guide the process of data-column extraction discussed in the next section. The specification of Label Cells is as follows:

²While the scan is vertical, the features used contain “horizontal” information

Input: a vector of binary features $F_{ij} = [f_0, \dots, f_k]$, where f_0, f_1, \dots, f_k are cell text, layout, text, content and context features, for each cell c_{ij} with column index i and row index j .

Output: a label $l_{ij} \in \{CH, DS, DC, DE, NDC\}$ for each c_{ij} .

Function: Assign a label l_{ij} to cell c_{ij} based on F_{ij} and the observed training dataset.

Cell Features. Chen et al. [18] describe a combination of layout and textual features they used in Frame Finder, a tool for locating attribute and value regions in spreadsheets. In addition to reusing some of those features, we also developed a comprehensive set of layout, text, content, and context features. (See Appendix A for the full list of cell features.) The development of these features involved 1) testing several combinations of feature sets, 2) evaluating the labeling performance and efficiency (computation time) of each feature set, and 3) identifying the set that provides the best balance between performance and efficiency.

Layout Features: Layout formatting provides valuable indicators about the type of cell being observed. For instance, the use of underline, bold, text alignment (e.g. center alignment), or merged cells is a strong indicator of a column-header cell.

Text Features: Analyzing the textual content of a cell also conveys important information about its type. Features such as *is_all_alpha* (does a cell contain only alphabet characters?), *is_all_numeric* (does a cell contain only numeric characters?), *is_empty_cell* (is the cell empty?) can help distinguish among column headers, data, and non-data cells.

Content Features: Understanding the possible meaning of a cell's content could provide indications about its position in a data column. For example,

in_year_range (does the cell contain a number between 1900 and 2050?) and *is_in_headers* (is the cell content has a word in the list of common column-header words?) may indicate that a cell contains a column header.

Context Features: Analyzing the features of a cell’s left, right, above and below neighbors can help in determining its label. For instance, features such as *is_above_num* (Does the cell above contain numbers only?) and *is_below_num* (Does the cell below contain numbers only?) can help determine whether a cell belongs to a data column or not. Our initial CRF model considered two neighbouring cells, but the current (improved) version considers four neighbouring cells in all directions.

CRF Training.

We collected a set of real and synthetic spreadsheets to use in training our CRF. This set covered various data-column configurations in a sheet, such as vertically stacked columns and data columns with various layout formatting (e.g., a column with a header that is two cells apart from the first data cell, a column with a merged-cells header, columns with no headers). To speed up the construction of the training set, we used a bootstrapping approach where we manually labeled an initial set and used it to train the CRF. Once an initial CRF model was available, we used it to quickly classify the cells of the rest of the set and then manually correct them as needed. We describe the construction of the training set in more detail below.

First, we divided the training set into an initial set and a remaining set. Second, using our C# implementation of Extract Cell Features, we extracted cell features from the initial set. Third, as shown in Figure 4.2, we manually labeled each cell in the initial set with one of the labels CH, DS, DC, DE, and NDC based on its position in a column. Fourth, using the CRF++ library [48], we tested our CRF

The screenshot shows the CRFResults interface with a spreadsheet. The spreadsheet has columns labeled 1 through 9. The data rows are as follows:

1	2	3	4	5	6	7	8	9
Sample #	water/sediment	date		um	Ø	Time (local)	Time PST	
S.236	water	17-Sep-13	DNA	0.2	25	23:13	22:13	
S.238	water	18-Sep-13	DNA	0.2	25	0:28	23:28	
S.240	water	18-Sep-13	DNA	0.2	25	14:07	13:07	
S.242	water	18-Sep-13	DNA	0.2	25	17:19	16:19	
S.244	water	18-Sep-13	DNA	0.2	25	18:05	17:05	
S.246	water	18-Sep-13	DNA	0.2	25	19:07	18:07	
Time PST	saturn03.1300.R...					Time PST	saturn03.1300.R...	
'2013-09-17 22:1...	3.42					'2013-09-17 23:2...	2.25	
'2013-09-17 22:1...	3.45					'2013-09-17 23:2...	2.26	
'2013-09-17 22:1...						'2013-09-17 23:2...	2.26	
'2013-09-17 22:1...	2.96					'2013-09-17 23:2...	2.24	
'2013-09-17 22:1...	2.91					'2013-09-17 23:2...	2.25	
'2013-09-17 22:1...	2.89					'2013-09-17 23:2...	2.24	
'2013-09-17 22:1...	2.87					'2013-09-17 23:2...	2.24	
'2013-09-17 22:1...	2.87					'2013-09-17 23:2...	2.24	
'2013-09-17 22:1...	2.86					'2013-09-17 23:2...	2.23	
'2013-09-17 22:1...	2.83					'2013-09-17 23:2...	2.23	
'2013-09-17 22:1...	2.8					'2013-09-17 23:2...	2.23	
'2013-09-17 22:1...	2.81					'2013-09-17 23:2...	2.23	
'2013-09-17 22:1...	2.85					'2013-09-17 23:2...	2.24	
'2013-09-17 22:1...	2.92					'2013-09-17 23:2...	2.25	
'2013-09-17 22:1...	2.95					'2013-09-17 23:2...	2.24	
'2013-09-17 22:1...	2.88					'2013-09-17 23:2...	2.24	
'2013-09-17 22:1...	3.38					'2013-09-17 23:2...	2.24	

A callout box labeled "Mislabeled cells" points to the cell containing "18-Sep-13" in the date column for sample S.246.

Figure 4.3: Example Label Cells results

model on a subset of the remainder set to determine labeling errors. Fifth, we used ReDiscover’s CRF_Results interface (Figure 4.3), to correct labeling errors in the new spreadsheets and then added the feature vectors and labels of these corrected spreadsheets to the training dataset. Finally, we repeated the fourth and fifth step with additional spreadsheets until we obtained satisfactory labeling results.

The Label Cells Results interface, shown in Figure 4.3, displays the output of the Label Cells CRF classifier. To inspect for labeling errors, a user first selects a sheet from the tree-view menu. Then, ReDiscover loads the selected sheet’s data into the table (in the middle of the figure), and colors each cell based on the label that the CRF classifier has assigned to it (yellow for column headers, light blue for data start, light green for data continuation, dark green for data end, and pink for

non-data cells). The user can select mislabeled cells and choose the correct label from the Edit Cells list (top), then save changes.

For example, spreadsheet Column 2 in Figure 4.3 actually contains two dataset columns: *water/sediment* and *saturn03.1300.R*. But because *water/sediment* (the top column) had text data, the CRF classifier captured the column header of *saturn03.1300.R* (the bottom column) as a data value. Consequently, the CRF classifier did not recognize the second column, which caused it to mistakenly label Cells 7 to 11 as DC. The user can correct the labels to match those in Column 1.

After correcting any labeling errors, the user can click *Retrain CRF* to add the corrected worksheet data to the training dataset to generate a new CRF model.

The output of Label Cells is a list of cell ids (e.g. *TD100.xls_Sheet1_[2,1]* belongs to the file *TD100.xls* and is located in Column 2, Row 1, in Sheet 1) and their assigned labels. ReDiscover passes this list to the Column Extractor, which uses it to determine column boundaries. We discuss Cell Label performance in Section 5.2.

4.2.2 Extract Columns

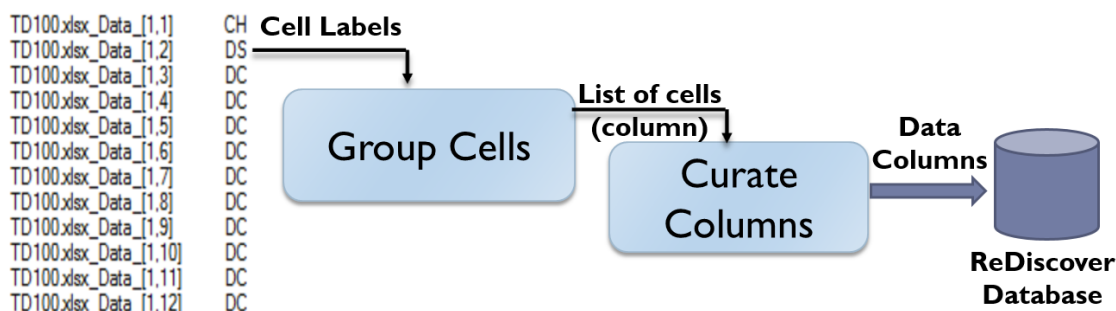


Figure 4.4: Extract Columns

Once Label Cells labels spreadsheet cells as part of a data column or not, the Extract Columns process uses these labels to guide data-column extraction. Figure 4.4 depicts the process of Extract Columns, which involves two steps: *Group*

Cells and *Curate Columns*. The first step, Group Cells, scans cell labels vertically looking for a group of cells that constitutes a column. In its simplest form, a column consists of zero or more CH cell, one DS cell, zero or more DC cells, and one DE cell, as shown in Figure 4.5(a). Group Cells can detect columns with various layout formats, some of which are presented in Figure 4.5: (a) a single data column, (b) vertically stacked columns, (c) columns with multiple header cells, (d) columns with spaces between column header cells and data cells, (e) columns with no column header, and (g) columns with discontinuous data cells.

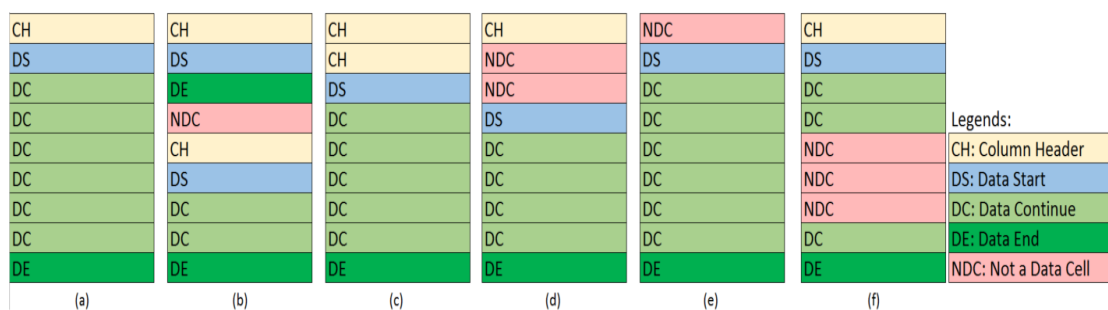


Figure 4.5: Column Layout Examples

In Group Cells, ReDiscover stores the cells of each identified column in a list, which is ordered by the original cells' positions in that column. Then, it collects column metadata, such as column order (the original column position in the dataset) and the addresses of the start and the end cells. Next, ReDiscover passes the resulting column lists and their metadata to *Curate Columns* for further column-extraction enhancement.

The second step, Curate Columns, analyzes the data values of each column to detect their types. In order to identify the data type of a column, ReDiscover must reconcile any type inconsistency among column values. For instance, in Figure 4.6, the type of columns *Size* and *Height* are float and integer respectively. However, as we discussed in Section 4.1.3, identifying the correct type in such cases is a challenging task.

Size	Height
NA	50
13.2622	No Reading
132.4336	340
135.215	81
381.24	68
NA	53
893.259	
902.3704	60
"905.12"	NA
1068.3942	65

Figure 4.6: An example of column data-type inconsistency in spreadsheets

The process of Type Inference starts by detecting the type of each individual value in the column. If all the values agree on the same type, then ReDiscover assigns that type to the column. Otherwise, it attempts to convert the values with distinctive types to either the type of the majority of values in the column or to null. For example, for the Size column in Figure 4.6, ReDiscover finds that ‘NA’ and ‘“905.12”’ are the two values with distinctive type (string). It looks up these values in the default-null list, a dictionary of words that are commonly used to refer to null values. ReDiscover converts the ‘NA’ to a null as it finds it the default-null list. For the ‘“905.12”’ value, ReDiscover attempt to convert that value to float, since it is the type of the majority of values in that column. However, if the conversion fails, ReDiscover replaces the value with null in case the number of distinctive values in the column is less than five percent. Otherwise, it converts the type of the whole column to *varchar* (string).

column_id	col_type	row_count	unique	Null	max_val	min_val	mean	Std_dev	common_val_0	cv0_freq
[dbo].[2_TD100.xlsx_Data_2_55][Sample Name]	varchar ...	54	18	0	6 mo 3	DM 1			1 mo 1	3
[dbo].[3_TD100.xlsx_Data_2_55][Z-Ave (d.nm)]	int ...	54	54	0	163	110	136	15.7321	110	1
[dbo].[4_TD100.xlsx_Data_2_55][Pd]	real ...	54	54	0	0.57	0.04	0.305	0.157321	0.04	1
[dbo].[5_TD100.xlsx_Data_2_55][Mean Count ...]	real ...	54	54	0	146.5	120	133.25	7.96607	120	1
[dbo].[6_TD100.xlsx_Data_2_55][Cumulants Fit...]	real ...	54	54	0	0.00095	0.000632	0.000791	9.43928E-05	0.000632	1
[dbo].[7_TD100.xlsx_Data_2_55][Measurement...]	varchar ...	54	54	0	Monday...	Friday, May ...			Friday, April 1...	1
[dbo].[8_TD100.xlsx_Data_2_55][T (°C)]	int ...	54	1	0	22	22	22	0	22	54
[dbo].[9_TD100.xlsx_Data_2_55][In Range]	real ...	54	1	0	98.4	98.4	98.4	0	98.4	54

Figure 4.7: Example column summaries

4.2.3 Compute Column Summaries

For ReDiscover to match columns and predict relationships between datasets without having to extensively compare column data, it applies data profiling to compute column summaries. As shown in Figure 4.7, for each column, ReDiscover collects information such as column data type, value range, common values and their frequencies, mean, variance, and the count of unique and null values. (See Appendix B, Table 1 for the full list of column summaries.) We discuss the use of column summaries in more detail in the next two subsections.

To aid in the relationship-prediction task, ReDiscover also collects spreadsheet metadata, such as file name, title, size, folder path, owner name, last saved by (user), date created, and date modified (See Table 4.1 for the full list of metadata.) Such metadata can prove useful, especially in resolving prediction conflicts when the computed column summaries and column correspondence are compatible with more than one relationship. For instance, suppose that dataset A fully corresponds to B , and the row count for both datasets is equal. Based on this information, there are two possible relationships: duplicate or template. ReDiscover can use spreadsheet metadata to help resolve such conflicts. For example, if *Author*, *WorkbookSize*, and *NumberOfSheets* metadata of A and B match, A and B are likely duplicates. Otherwise, it is more likely that A and B share the same

Table 4.1: Description of spreadsheet metadata that ReDiscover collects.

Metadata Name	Description
<i>Author</i>	The author name of the specified document
<i>Company</i>	The company name
<i>Title</i>	The title of the spreadsheet document
<i>Subject</i>	A brief description about the subject of the specified document
<i>Comments</i>	A brief description about the specified document
<i>Path</i>	The file name and location of the specified document
<i>NumOfSheets</i>	The number of sheets in a spreadsheet
<i>WorkbookSize</i>	Spreadsheet document size in bytes
<i>DateCreated</i>	The date and time that the specified document was created
<i>DateLastModified</i>	The date and time that the specified document was last modified
<i>DateAccessed</i>	The date and time that the specified document was last accessed
<i>LastSavedBy</i>	The user name of the person who last saved the spreadsheet

template.

4.2.4 Match Columns

In ReConnect, the *Correspond Columns* process (Section 3.1.2) uses only column names to match columns and relies on the user to confirm or correct the resulting mapping. However, since ReDiscover has to match columns between many dataset

pairs in a collection, it is infeasible to ask for the user's help on every pair. We apply a supervised learning model, namely *support vector machines* (SVMs) [20] to improve the column-matching process, and hence reduce user involvement. The goal of Match Columns is to find the best column correspondence between each pair of spreadsheets. In order to apply SVMs to column matching, we first need to formulate it as a classification task

Column Matching as a Binary Classification Problem To determine the column correspondence between a pair of datasets A and B , we want to find the maximal set M of matching column pairs. A pair of columns (a, b) match if the columns are semantically related and they describe the same real-world object. We denote a matching column pair by $(a, b) \in M$, and refer to it as the mapping of a to b . The *column correspondence* between A and B is a set M of mapped column pairs from A and B . We can formalize the problem of identifying pairs of matching column as a binary classification problem with the following specifications.

Input: The similarity vector v for a column pair (a, b) in $col(A) \times col(B)$.

Output: 1 for matching column pairs and -1 for non-matching pairs.

Function: Assign the column pair (a, b) to class 1 (matching) or to class -1 (non-matching) based on its similarity vector v and the observed training dataset.

The Similarity Vector of a Column Pair. As shown in Figure 4.9, for each pair of columns (a, b) , ReDiscover uses the columns summaries generated by the Compute Column Summaries process to derive a similarity vector v that indicates similarity or dissimilarity between each summary component of the two columns.

column_id	col_type	row_count	unique	Null	max_val	min_val	mean	Std_dev	common_val_0	cv0_freq	common_val_1	cv1_freq	...	
Sample	int	...	2	2	0	1351	1350	1350	0.707...	1350	1	1351	1	...
Summary of column Sample of Spreadsheet A														
column_id	col_type	row_count	unique	Null	max_val	min_val	mean	Std_dev	common_val_0	cv0_freq	common_val_1	cv1_freq	...	
Sample #	int	...	4	4	0	1353	1350	1351	1.29099	1350	1	1351	1	...
Summary of column Sample # of Spreadsheet B														
Column Name	Type	Unique Count	Null Count	Max Value	Min Value	Mean	Std Deviation	Common Value 0 (CV0)	CV0 Frequency	...				
0.75	1	0.5	1	0.99	1	0.99	0.55	1	1	...				
The similarity vector of columns A.[Sample] and B.[Sample #]														

Figure 4.8: An example of two column summaries and their similarity vector

Each similarity vector consists of p similarity metrics (currently $p = 18$). Each metric function $s_j(a, b)$ ($j = 1, \dots, p$), computes a similarity score between two corresponding column-summary components. This score ranges between zero (incompatible component values) and one (identical component values). For example, *Unique* value count (2) of column *Sample* (Figure 4.8(a)) and the *Unique* count (4) of column *Sample #* (Figure 4.8(b)) have a similarity score of 0.5, as shown in Figure 4.8(c).

We use different similarity metrics for different components of the column summary. For string components, such as column names and common values for categorical columns, we used Levenshtein distance [50], which is the minimum number single-char edits required to change one word into the other (normalized between 0 and 1.) For numerical summaries, we simply compute the scaled difference (D) between two numerical values ($D = \left| \frac{(X_1 - X_2)}{((X_1 + X_2)/2)} \right|$).

As we discussed in Section 2.3.2, simple count statistics do not provide good summaries for computing similarity between groups of categorical values. There are several similarity metrics, such as *Jaccard coefficient* and Levenshtein distance, that are often used to compare groups of categorical columns. However, it is quite expensive to compute such metrics for many column pairs and large datasets. Thus, we developed a technique based on *Bloom filters* that enables us to inexpensively

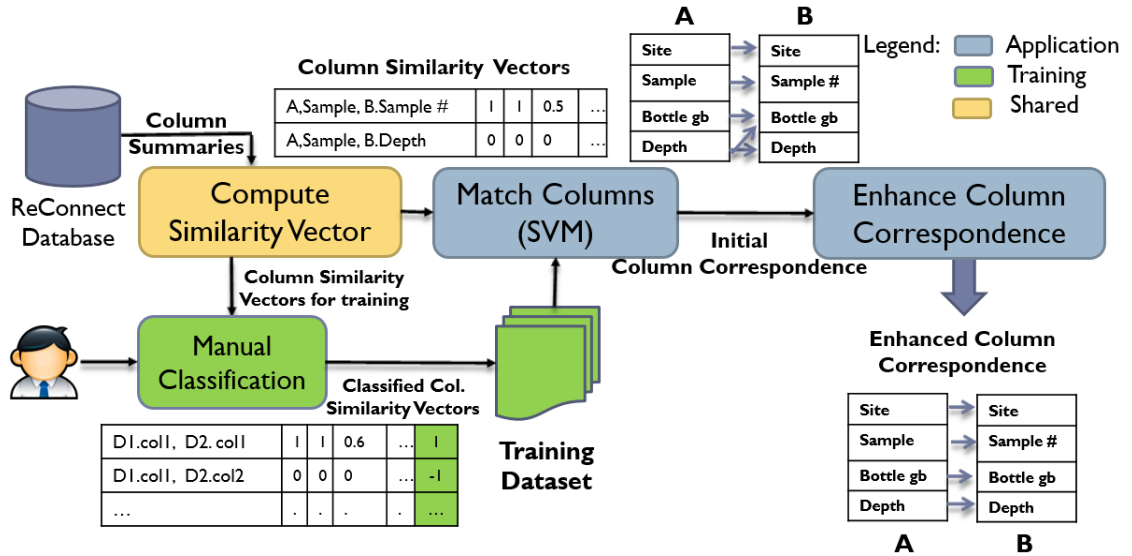


Figure 4.9: The Match-Column process

approximate similarity between categorical columns.

Computing Set Similarity using Bloom Filters

In our approach, we individually compute a fixed-size bit-vector representation of each data column. Pairwise column comparisons then use these bit-vectors rather than the complete set of column values. For instance, for column matching, Match Columns computes similarity between categorical columns by estimating the *Dice coefficient* [28] between the resulting bit-vector pairs. The Dice coefficient of sets X and Y is $\frac{2 \times |X \cap Y|}{(|X| + |Y|)}$ and can be approximated from the number of 1-bits in the bitwise *and* of the filter vectors [65]. Predict Relationships also uses Bloom filters in computing features for relationship prediction (Section 4.2.5).

With Bloom filters, false positives are possible but false negatives are not. To achieve a given expected false-positive rate p , the size m of a Bloom filter must be proportionate to n , $(-\frac{n \times \ln p}{(\ln 2)^2})$, where n is the number of elements in the set (column) [68]. Furthermore, Jain et al. [45] showed that when Bloom filters are used for measuring similarity between two sets, a 97% matching accuracy can be

achieved by setting the number of bits in the Bloom filter to $8n$. We can select n for a given collection of datasets by determining an upper bound on the number of rows in any dataset.

Based on the analysis of Jain et al., and the characteristics of our test collection, we chose a Bloom filter size of 512 bytes. Based on a random sample of columns, we find at this size the estimated Dice coefficient is always within 2% of the actual Dice coefficient.

Training the Match-Columns Classifier In Figure 4.9, we show both the Match Columns training and application phases. In the training phase, we compute similarity vectors for pairs of columns using the *Compute Similarity Vector* process, which is used by both phases. The spreadsheets we used for training include various column-matching scenarios, such as columns that are differently named but conceptually identical, columns with identical names but conceptually different, incompatible columns, and duplicate columns. Then, we manually labeled the column pairs as matching or non-matching. Next, using LIBSVM [17], a software library for SVMs, we trained the Match Columns classifier.

In the application phase, the *Compute Column Similarity* process first computes a similarity vector for each pair of columns. The Match Columns classifier first analyzes each similarity vector and classifies it as matching or non-matching. Second, ReDiscover forms the set (M) of all column pairs (a,b) that are labeled as matching (i.e, a column correspondence). Third, it passes M to the *Enhance Column Correspondence* process, which ensures that all the column mappings in M are one-to-one, that is, no column in A maps to more than one column in B and vice versa. We describe the *Enhance Column Correspondence* process in more detail below.

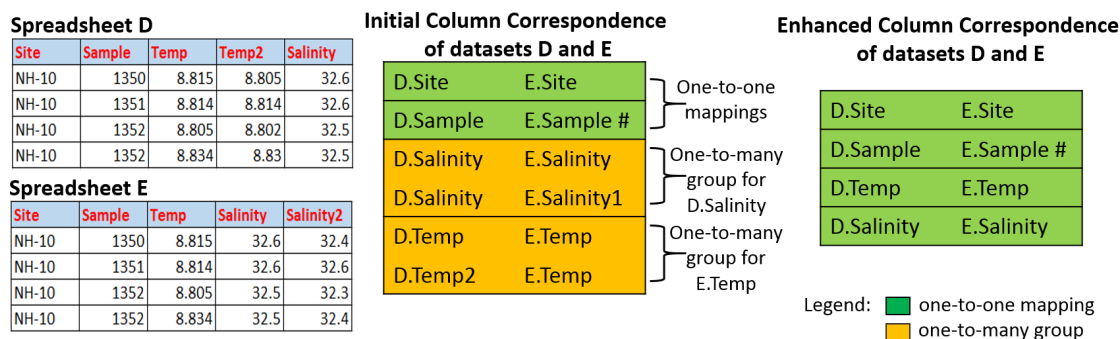


Figure 4.10: An example of enhancing the column correspondence of datasets D and E .

Enhance Column Correspondence Using our running example, suppose that Jennifer receives two new spreadsheets, D and E , which both are extended versions of spreadsheet C (Figure 2.1). D has an extra column, $Temp2$, that contains new temperature readings for the same samples in C . Spreadsheet E has an extra column, $salinity2$, which represents the water salinity readings for the same samples in C at a later time. As shown in Figure 4.10, when ReDiscover attempts to match the columns of spreadsheets D and E , the resulting initial column correspondence is ambiguous. Column $salinity$ of D is mapped to both columns $salinity$ and $salinity2$ of E and column $Temp$ of E is mapped to both columns $Temp$ and $Temp2$ of D .

For each column with ambiguous mappings, ReDiscover groups the column correspondences involving that column. For instance, for column $D.salinity$, it creates a *one-to-many* group that includes $D.salinity \leftrightarrow E.salinity$ and $D.salinity \leftrightarrow E.salinity2$ mappings. Similarly for column $E.Temp$, it creates another *one-to-many* group containing $E.Temp \leftrightarrow D.Temp$ and $E.Temp \leftrightarrow D.Temp2$ mappings. Each one-to-many group contains ambiguous mappings and only one mapping from such group should be selected. To overcome such matching ambiguity, we developed the *Enhance Column Correspondence* process (Figure 4.9), which we detail in Algorithm 3.

Algorithm 3 Enhance Column Correspondence algorithm

Input: initial column correspondence for table A and B (col_corr).

Output: enhanced column correspondence ($enhanced_col_corr$) that represents a one-to-one mapping.

1. $matching_score[] = getMatchingScores(col_corr)$; $\triangleright getMatchingScores()$ computes a matching score for each mapping in col_corr .
 2. $idx = 0$;
 3. **while** col_corr is not empty **do**
 4. **while** $idx < col_corr.count$ **do**
 5. $one_to_one = findO2O(col_corr)$; $\triangleright findO2O()$ finds the first one-to-one mapping in col_corr .
 6. $enhanced_col_corr.add(O2OMapping)$;
 7. $col_corr.remove(O2OMapping)$;
 8. **end while**
 9. $one_to_many[] = findO2M(col_corr)$; $\triangleright findO2M()$ finds the one-to-many group with the least number of column involved.
 10. $bestO2M = getBestO2M(one_to_many[])$; $\triangleright getBestO2M()$ finds the mapping with the highest $matching_score$.
 11. $enhanced_col_corr.add(bestO2M)$;
 12. $col_corr.remove(bestO2M)$;
 13. $col_corr = reduce_corr(bestO2M)$; $\triangleright reduce_corr()$ removes from col_corr the remaining mappings that are not selected in the 1-many group.
 14. **end while**
-

ReDiscover first searches for all one-to-one mappings ($D.Site \leftrightarrow E.Site$ and $D.Sample \leftrightarrow E.Sample \#$) and adds them to the enhanced column-correspondence list ($enhanced_col_corr$), and removes these mappings from the original column-correspondence list (col_corr). Next, it retrieves the similarity vectors of all column pairs in the remaining column correspondence set. Then, for each vector, it computes a *matching score*—the sum of all similarity metrics in that vector. For the remaining two one-to-many groups ($D.Salinity$ and $E.Temp$), ReDiscover chooses the group with the fewest columns involved. In our example in Figure 4.10, both of the one-to-many groups have two mappings each, so ReDiscover selects the group with the mapping that has the highest matching score (say the $D.Salinity$ group). From the selected group, ReDiscover adds the mapping with the highest matching score ($D.salinity \leftrightarrow E.salinity$) to $enhanced_col_corr$ and removes it from the col_corr list. Next, $refined_corr(bestO2M)$ discards any mapping that has $D.salinity$ from col_corr list. ReDiscover repeats this process with the remaining one-to-many groups ($E.Temp$) until col_corr is empty.

Finally, ReDiscover passes the enhanced column correspondence to the *Predict Relationships* process. Because relationships are classified by column correspondence (*Full Correspondence*, *Sub-Correspondence*, *Extension Correspondence*, and *No Correspondence*), ReDiscover can limit the number of relationships it investigates for a pair of spreadsheets using their correspondence classification.

4.2.5 Predict Relationships

The *Predict Relationships* algorithm is an improved version of *Suggest Relationships* in ReConnect [5]. While both algorithms share the use of column-correspondence types (Full-, Sub-, Extension, and No Correspondence) to limit the number of relationships to investigate, the *Predict Relationships* algorithm also uses an improved column-correspondence and spreadsheet metadata in predicting relationships. Additionally, it uses a set of SVM classifiers (one per relationship type) for relationship

prediction. We discuss the details of the Predict Relationships process below.

Predict Relationships Process

As shown in Algorithm 4, Predict Relationships takes as input column summaries, a column correspondence, and spreadsheet metadata for a pair of datasets. First, it identifies the type of column correspondence between two datasets to determine the set of possible relationships between them. Then, for each possible relationship, Predict Relationships 1) computes the relationship feature vector based on the pair’s column summaries, column correspondence, and metadata, and 2) sends the resulting feature vector to the relationship SVM classifier (Each relationship has a separate classifier.) Based on the feature vector and the observed training dataset, the relationship classifier assigns the dataset pair to either class 1 (relationship exists) or class -1 (relationship does not exist). The SVM classifier also returns a score that represents the relationship likelihood measure (*prediction score*). Finally, if the resulting prediction score is greater than or equal to a given threshold, α (currently $\alpha = 30$), then Predict Relationships adds the dataset pair, the predicted relationship, and the prediction score to the list of predicted relationships.

Computing Relationship Features

For each relationship, ReDiscover computes indicative features of a relationship from column summaries, such as min and max values, count of unique and null values, and mean and standard deviation. For example, for the *row-containment* relationship, ReDiscover uses the column statistic *min* value to compute *MinAWithinRangeB*, a binary feature that is set to “1” when the min value range of each column in *A* is within the value range of the column it is mapped to in *B*. Another example is the *RowCountDifference* feature that ReDiscover computes for the *complementation* relationship. This feature measures the similarity between

Algorithm 4 Predict Relationships algorithm

Input: A dataset pair $(D1, D2)$, column correspondences (col_corr); column summaries (col_summ) and spreadsheet metadata (SS_meta).

Output: List of pairs of dataset names, the predicted relationship, and the prediction score ($predicted_relts$).

1. $col_corr_type = identify_correspondence(Col_Corr);$ ▷
 $identify_correspondence()$ takes the column correspondences and returns Full Correspondence, Extension Corresponded, Sub-correspondence or No Correspondence
 2. $possible_relts = get_the_relationships\ associated\ with\ the\ col_corr_type$
 $type;$
 3. **for** each relationship r in $possible_relts$ **do**
 4. $features = compute_Features(col_corr, col_stats, SS_meta, r);$
 5. $Score = run_relt_SVM(r, features);$
 6. **if** $Score \geq \alpha$ **then**
 7. $predicted_relts.add(D1.name, D2.name, r, Score);$
 8. **end if**
 9. **end for**
-

the row counts of a column pair, and takes a value between [0,1] (0 means that row counts are very different, and 1 means they are identical.) Note that some relationships, and hence some features, are directed (e.g., A column-contains $B \neq B$ column-contains A).

Predict Relationships also uses Match Columns' results in computing indicative features for a relationship. For instance, it uses the likelihood score resulting from the SVM classifier for column matching to compute the average similarity between

all corresponding columns in A and B . Such an average similarity feature can be used in predicting the *duplicate* and the *containment* relationship as it measures the overall similarity between all corresponding columns between two datasets.

Spreadsheet metadata, such as file name, size, and folder path (see Section 4.2.3), can improve relationship-prediction accuracy. For example, knowing that a pair of datasets have the same file name, size, author, and last modified date provides evidence of a *duplicate* relationship. Another example is when the metadata of two datasets are similar except the size of one file is larger than the other. Predict Relationships uses the file size feature, among other features, in predicting the *containment* relationship.

Computing Relationship Features using Bloom Filters

We incorporated Bloom filters initially to help with column matching, but we also realized that they could be useful for relationship prediction. Predict Relationships uses Bloom filters in computing features for predicting relationships, such as *containment*, *duplicate*, *prefix*, and *suffix*. In the case of the *containment* relationship, it computes *isAllBVofAContainedInB*, a binary feature that is set to “1” when the bit vector of each column in dataset A is contained in the bit vector of the column it is mapped to in B , or zero otherwise. Predict Relationships tests if a bit vector bv_1 is contained in bv_2 by computing the bitwise *and* between bv_1 and bv_2 and testing if it matches bv_1 .

For the duplicate relationship, Predict Relationships computes the *avgDiceSimilarityforDuplicate* feature, which is a value between zero and one that represents the average Dice similarity estimates for all corresponding columns (computed in the Match Columns process) between two datasets. The higher the value of the *avgDiceSimilarityforDuplicate* feature is, the more similar the content of the two datasets are, the more likely they are identical.

Predict Relationships also uses Bloom filters to compute indicative features for

ordered relationships, including *prefix*, *suffix*, and *reordered rows*. When ReDiscover predicts that a dataset A is row-contained in dataset B , it further checks whether A could be a prefix (or suffix) of B by computing two Bloom filters for each data column in the containing dataset B : one for the first half of the column, B_1 , and one for the second half, B_2 . It also computes Bloom filters A_1 and A_2 for the first and second half of the corresponding column in A . If $A_1 \subseteq B_1$, it is evidence that A is a prefix of B . But if $A_2 \subseteq B_2$, it is evidence that A is a suffix of B .

When ReDiscover predicts that a pair of datasets, C and D , are equal, it also checks whether the reordered rows relationships holds between them as follows. First, it computes a *trigram* Bloom filter for each data column in the set of corresponding columns (M). Each element in this filter represents a contiguous ordered sequence of three values from the data column. Next, ReDiscover computes Dice similarity between the Bloom filters of each pair of corresponding columns. The less similar the Bloom filters of the corresponding columns in C and D are, the more likely that the reordered rows relationship holds between them.

Predict Relationships Results

Datasets	Predicted Relationship	Prediction Score
Species_Code.xls, Species_ids.xls	Row-Containment	90
05-25.xls, 003_4.xls	Template	75
SE_yr.xls, E_mo.xls	Duplicate	35

Figure 4.11: Example results of the *Predict Relationships* algorithm

As shown in Figure 4.11, Predict Relationship outputs a list of dataset names, the predicted relationship, and the prediction score. A scientist can quickly select a

pair of datasets that are most likely to have a relationship for further examination (with ReConnect, for example), without having to visually inspect all datasets in the collection. Ordering the list of predicted relationships by the prediction score allows users to work with the most reliable predictions first.

To conclude, we built ReDiscover with the goal of automating relationships discovery in collections of scientific datasets. ReDiscover is an end-to-end prototype system that helps scientists identify from a collection of datasets the pairs that are likely related and the predicted relationship between them. It applies CRFs to automate data-column extraction, computes column summaries using data profiling and approximate categorical column summaries using Bloom filters, and uses SVMs to automate the process of column-matching and relationship-prediction between dataset pairs. It also uses Bloom filters to compute indicative relationship features for relationships such as, prefix, suffix, and reordered rows. In the next chapter, we discuss in detail the results of our experimental evaluation of ReDiscover.

CHAPTER 5: EXPERIMENTAL EVALUATION

We developed an end-to-end prototype of our relationship-prediction approach, to guide the design process and to evaluate the feasibility of our approach. After we developed the first version of ReDiscover, we performed an initial investigation to identify the processes whose improvement would benefit its prediction performance the most (Section 5.1). We found that the quality of the Label Cells process has a significant effect on the quality of downstream processes, such as Extract Columns and Compute Column Summaries. We also identified a number of performance issues with the Label Cells and Match Column processes that affected ReDiscover’s overall performance.

Based on the results of this investigation, we first implemented several improvements to Label Cells, including fixing cell-extraction errors and improving its CRF model. To evaluate the improved version of the Label Cells process, we conducted a second experiment, which we discuss in detail in Section 5.2. Second, we developed a new approach that uses Bloom filters for computing fast approximate summaries for categorical columns to improve column matching for such columns. We also used Bloom filters for computing indicative relationship features (see Section 4.2.5). Finally, we evaluated the updated version of ReDiscover on selected relationships to assess its overall relationship-prediction accuracy (Section 5.3).

5.1 PRELIMINARY EVALUATION

The goal of this experiment is to assess the effect of the result quality of different components on the results of later processes, and identify the processes whose

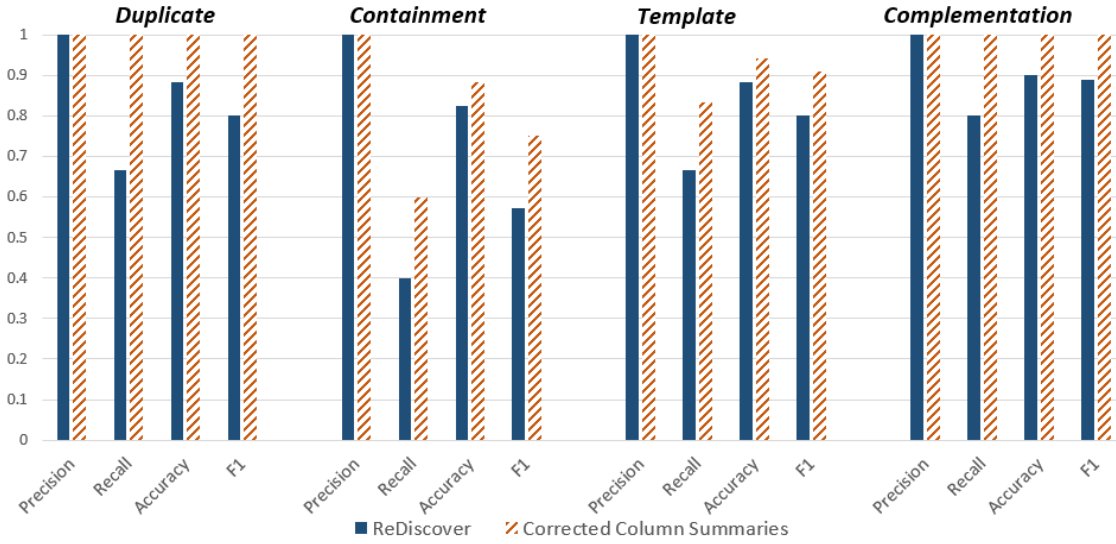


Figure 5.1: Relationship prediction performance results of ReDiscover and ReDiscoverMCS

improvement would benefit overall prediction performance the most. In this section, we report the results of our initial evaluation of ReDiscover on a variety of real-world and synthetic spreadsheets. We discuss our evaluation methodology in detail below.

5.1.1 Methodology

For our experiment, we used 10 pairs of real-world spreadsheets that we collected from our research collaborators, and 10 pairs from the EUSES corpus [33]. We modified the EUSES spreadsheets to construct relationships and features that are not covered by the first set. We used ReConnect to identify relationships between all pairs as a baseline for judging ReDiscover. Our evaluation methodology consists of three parts.

Part 1: Predicting Relationships using ReDiscover

To evaluate the performance of the initial implementation of ReDiscover on predicting relationships, we first used it to extract columns and compute their summaries, identify column correspondence, and predict the relationship of each pair of spreadsheets in our test set. Then, we compared ReDiscover’s relationship predictions with the previously identified relationships and computed the precision, recall, accuracy, and F1 score of those predictions.

Part 2: Predicting Relationships using Actual Column Summaries

To study the effect of result quality of ReDiscover’s individual processes on the result quality of subsequent processes, and to identify the processes that when improved would improve overall prediction accuracy, we evaluated a variant of ReDiscover, which we call ReDiscoverMCS, in two steps. In the first step, we manually computed column summaries for each spreadsheet of our test set. These error-free statistics serve as the *ground truth* column summaries. Next, we fed ReDiscover’s Extract Columns process the correct cell labels for the tested spreadsheets, and compared the resulting column summaries with ground-truth summaries. The purpose of this step was to analyze the effect of cell-labeling errors on the result quality of the Extract Columns, Match Columns and Predict Relationships processes.

In the second step, we used ReDiscover to compute the column correspondence and predict relationships for each pair using these manually computed summaries. For each tested pair, we recorded the resulting column correspondence and the predicted relationship. Lastly, we evaluated the accuracy of ReDiscoverMCS by computing the precision, recall, accuracy, and F1 score of its predictions.

Part 3: Evaluating ReDiscover’s Prediction Score

This part is separate from the previous two parts, and focuses on evaluating ReDiscover’s prediction score. We analyzed ReDiscover’s prediction scores with pairs

that are related and those that are not to evaluate whether they provide a useful guide to the user about likelihood of a relationship. For each relationship, we first selected a subset of dataset pairs that have a specific set of relationships and used ReDiscover to predict relationships between these pairs. We then computed the average prediction score for the entire set. Second, we repeated the previous step for the same relationship with other dataset pairs lacking that relationship. Finally, we computed the overall average of prediction scores for pairs that had the relationship, and the overall average for pairs that do not have the relationship.

5.1.2 Results

Figure 5.1 shows the precision, recall, accuracy, and F1 score of ReDiscover and ReDiscoverMCS predictions of *duplicate*, *row-containment*, *template*, and *complementation* relationships (Part 1 and 2). Precision is the number of positive relationship predictions that are correct, recall is the percentage of positive-labeled relationships that were predicted as positive, accuracy is the percentage of predictions that are correct, and F1 score is the weighted average of precision and recall ($2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$).

For the *duplicate* and *complementation* relationships, ReDiscoverMCS accurately predicted them in all spreadsheet pairs, whereas ReDiscover only predicted 66% of the tested pairs with 0.80 F1 for the *duplicate* relationship (80% with 0.88 F1 for the *complementation* relationship). Because of cell-labeling and column-extraction errors, ReDiscover computed inaccurate column summaries, which resulted in mismatching several columns. Consequently, ReDiscover was not able to predict the *duplicate* and *complementation* relationships for pairs with inaccurate column summaries. On the other hand, ReDiscoverMCS did perfectly well with various spreadsheet pairs for these relationships because it had accurate column summaries.

ReDiscoverMCS outperformed ReDiscover by 20% in predicting the *row-containment* relationship with an F1 of 0.75. However, they both fail to predict this relationship in few pairs of spreadsheets. When we analyzed the results of Part 1 and Part 2 for these pairs, we noticed that, even with manually computed columns summaries, ReDiscover mismatched several categorical columns. As a result, ReDiscover categorized the column correspondences for these pairs as *sub-correspondence*, where it should have categorized them as *full correspondence*.

Regarding the *template* relationship, ReDiscoverMCS correctly predicted 83% of the pairs, while ReDiscover predicted only 66% of them. ReDiscoverMCS's F1 was 0.90 and ReDiscover's was 0.80. As was the case with predicting the duplicate relationship, errors in cell labeling, which led to errors in column extraction and column matching, affected the accuracy of ReDiscover's predictions. ReDiscoverMCS also missed a few pairs because of inaccurate column matching.

With respect to the results of the first step of Part 2, we found that the extracted column summaries, which are based on correct cell labels, matched those that were computed manually (i.e., ground truth summaries). Consequently, the low performance of ReDiscover compared to ReDiscoverMCS was a result of incorrect labeling by the Label Cells process of ReDiscover. Thus, we concluded that the result quality of ReDiscover's Label Cells process has a significant effect on the result quality of the downstream processes.

Figure 5.2 shows the results of evaluating ReDiscover's prediction score (Part 3). ReDiscover predicted the presence of the *duplicate*, *containment*, *template*, and *complementation* relationships between pairs where these relationships hold with an overall average prediction score of 75.5. It also predicted the absence of these relationships between pairs lacking the relationship with an overall average prediction scores of 3.9. We conclude from these results that ReDiscover's prediction score is a useful guide for identifying the pairs that are most likely related. Users

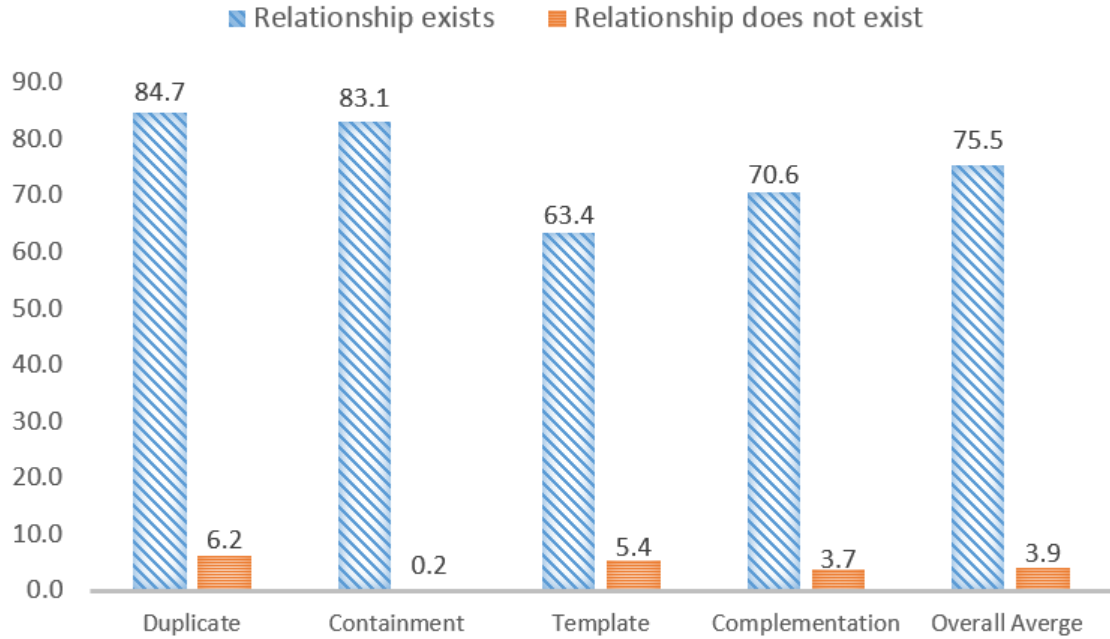


Figure 5.2: ReDiscover’s Average Prediction Score

can save time by skipping pairs with low prediction score as they are unlikely related. Additionally, by starting with pairs with high prediction scores, users will not waste time with false-positive predictions.

5.1.3 Discussion

Regarding whether improving certain stages of ReDiscover would improve its prediction performance, we concluded from our assessment results (Part 1 and 2) that with accurate data-column extraction, ReDiscover produces accurate column summaries, and hence derives better column correspondences and predicts relationships more accurately. With the presence of cell-labeling errors, such as labeling data values as NDC or labeling a column’s last cell as DC, the accuracy of ReDiscover’s column summaries drops. For instance, because of labeling several non-data cells as DC, ReDiscover incorrectly inferred that a numerical column type was *string*;

thus, it did not compute numerical summaries, such as mean and standard deviation. Improving the Label Cells process should yield better column summaries, hence enhancing the prediction quality.

The main classification errors of Label Cells were false positives on DC, false negatives on DE, and false negatives on NDC. Based on these observations, we made several improvements to the Label Cells process, and its CRF model. First, we identified and fixed a number of cell-feature extraction bugs. The classification accuracy of the CRFs model depends on the accuracy of the extracted cell features. For instance, we found that bugs in *is_empty* and *is_in_nulls* feature-extraction routines were responsible for most of the false negatives on NDC.

Second, to handle the false positives on DC and the false negatives on DE classification errors, we updated our CRF Model to consider further levels of neighboring cells in both direction to improve cell labeling accuracy. The initial CRF model considered only the two adjacent neighboring cells in both directions when it is classifying a given cell. Consequently, the model was not able to identify the end cell (DE) of a data column accurately, and labeled several DE cells as data continuation (DC). We extended our CRF model to consider four neighboring cells in both directions. As a result, this improved version of the Label Cells process was able to identify DC and DE cells more accurately, as we will see in the next section.

The results of our preliminary evaluation also show that because the column statistics of the first version of ReDiscover did not provide much insight for categorical columns, the Match Columns process produced inaccurate column correspondences, which reduced the accuracy of ReDiscover’s relationship prediction. Based on these observations, we developed a new approach that uses Bloom filters to compute bounded-size column summaries to approximate the values of a categorical column inexpensively and effectively (see Section 4.2.4 and 4.2.5). We discuss our evaluation of the improved version of ReDiscover in Section 5.3.

In conclusion, the results of our preliminary evaluation suggest that the accuracy of ReDiscover’s predictions were promising. We also found that by improving Label Cells and Match Columns processes we can improve the overall performance of our relationship-prediction approach. In the next section, we discuss our evaluation of the improved version of Label Cells process.

5.2 LABEL CELLS EXPERIMENT

Our initial investigation revealed that the performance quality of Extract Columns, Compute Column Summaries, and Match Columns processes depends significantly on the quality of the cell-labeling process. After implementing several improvements to the initial Label Cells process, we evaluated it to assess its accuracy, and also to see whether the use of a sophisticated machine learning technique, such as CRFs, is justified for such a task (or if a simpler labeling scheme would suffice).

In this experiment, we used 23 real-world spreadsheets that we collected from our research collaborators and from the EUSES corpus [33]. The cells of our test spreadsheets were labeled by a human expert using the Label Cells Results interface (Figure 4.4) to give ground truth. Our evaluation methodology is as follows.

5.2.1 Methodology

First, to assess the performance quality of the Label Cells process, we evaluated its CRF classifier (LC-CRF, described in Section 4.2.1). We used LC-CRF to label the cells of our test data³, and then we compared LC-CRF labels with the human-expert labels and computed the precision, recall, and F1 score of Label Cells’s classifications. We evaluated per label, because some cell types are more frequent and we did not want them to dominate the results.

Second, to answer the question of whether such a sophisticated CRF model is

³Test data were separate from the data used to train LC-CRF

needed for the cell labeling task, we evaluated a restricted version (CRF-Base) of our CRF model. CRF-Base does not use the left and right content features (see Appendix A, Table ??), and it considers only the features of cells the immediately above and below of the cell it is classifying. In contrast, LC-CRF uses the right and left content features, and it considers four level of neighboring cells in both directions.

5.2.2 Results and Discussion

		Precision	Recall	F1-Score
NDC	CRF-LC	0.92	0.97	0.93
	CRF-Base	0.88	0.84	0.85
CH	CRF-LC	0.78	0.86	0.80
	CRF-Base	0.56	0.73	0.59
DS	CRF-LC	0.71	0.81	0.74
	CRF-Base	0.91	0.59	0.70
DC	CRF-LC	0.93	0.90	0.90
	CRF-Base	0.76	0.88	0.80
DE	CRF-LC	0.75	0.88	0.79
	CRF-Base	0.58	0.81	0.65

Table 5.1: Label Cells Performance.

As shown in Table 5.1, the majority of CRF-LC’s performance metrics, including precision, recall, and F1-score, are markedly better than those of CRF-Base for all types of labels. The only exception is that CRF-Base’s precision (0.91) for the data-start label (DS) is better than that of CRF-LC (0.71). However, we also notice that for the same label, CRF-LC’s recall (0.81) is considerably better than

that of CRF-Base (0.59). When we investigated the high precision of CRF-Base, we found that it generally tends to generate very few DS labels, which leads to a low rate of false positives and a high rate of false negatives, and hence CRF-Base has high precision and low recall. On the other hand, CRF-LC generated more DS labels and made some errors (false positives) but it identified most of the DS labels.

When extracting a data column, it is important to correctly identify its header (CH), the data (DC) and the end (DE) of that column. CRF-LC exhibits high performance in classifying CH, DC and DE, with average F1-scores of 0.80, 0.90 and 0.79 respectively. The ability of CRF-LC to use the left and right content features and its ability to consider the features of several cells above and below the cell it is classifying improves its labeling performance. Overall, we conclude from Table 5.1 that CRF-LC predicts each type of label with good accuracy, and that by comparing CRF-LC to the baseline method, CRF-Base, we find that the use of our more sophisticated CRF model is justified for our cell-labeling task. Based on the results of our initial investigation (Section 5.1), we know that this difference in accuracy makes a notable difference in the performance of the downstream processes, such as Extract Columns and Match Columns. In the next section, we evaluate the overall performance of ReDiscover in predicting relationships between datasets.

5.3 PREDICT-RELATIONSHIPS EXPERIMENT

In this experiment, we evaluate the overall relationship-prediction effectiveness of the improved version of ReDiscover, which 1) has a better-tuned CRF for cell labeling, 2) uses Bloom filters for both approximating categorical column summaries and computing indicative relationship features, and 3) uses improved SVM models that are trained on new features for relationship prediction. Our assessment of prediction performance is centered on the benefit to the user. Suppose we order

pairs of datasets using our prediction scores, and have the user work his or her way down the a ranked list of results, confirming or rejecting the predicted relationships (say, using ReConnect). We can compare the work a user would need to do using our order to that required with another order by considering essentially recall and precision:

Q1 How far down the list would a user need to go to find a given fraction of the true relationships?

Q2 How many false predictions would the user have seen by that point?

In addition, we want to know if the absolute prediction score can provide a cutoff threshold for the pairs the user needs to consider:

Q3 Is there a strong correlation between ReDiscover’s prediction score and the likelihood of a relationship?

5.3.1 Methodology

We evaluated ReDiscover on five relationships: *duplicate*, *row-containment*, *template complementation*, and *reordered-rows*. We used five test sets for our preliminary evaluation, a collection of 25 spreadsheets from EUSES, a collection of 80 CSV files produced by wearable activity monitoring devices, a collection of 20 datasets related to a geology research project, and two datasets of 21 and 22 spreadsheets, which we modified from EUSES to construct relationships and features that were not covered by the first three sets. For each test set, we went through all pairs individually to identify the existing relationships in the collection. Our relationship prediction evaluation methodology has three parts:

Part 1: Predicting Relationships using ReDiscover

To measure ReDiscover’s ability to find all relevant relationships (Q1), and to

quantify wasted user effort resulting from incorrect predictions (Q2), we used ReDiscover to score the relationship of each pair of spreadsheets in our five test sets, and ranked the results based on the prediction score. Then, we compared these predictions with manually identified relationships. We computed the precision at standard recall levels (discussed below), and the average precision over all relevant predicted relationships.

Part 2: Predicting Relationships using an approximate Human Baseline Approach

We wanted to know if ReDiscover’s predictions were better than what a human might do by “eyeballing” the spreadsheets in a collection. To approximate human performance, we polled spreadsheet users as to what strategies they might use to make an initial assessment of whether two spreadsheets are related and how. For example, some indicated that they would look at spreadsheets’ metadata (e.g., file name, size, author, creation date) for hints as to whether the duplicate relationship might hold. Other users stated that they would examine whether the column names and data types of a dataset pair match in order to identify whether they share the same template. We built a predictor based on SVMs that uses such strategies and features, for instance *columnNamesMatch*, *isDataTypesMatch* and *isColumnOrderMatch*, in classifying relationships, which we call the Human-Baseline Approach (HBA). (See Appendix C for the complete description of HBA’s features.)

We judged the prediction for precision of Part 1 and 2 based on what are the most informative relationships. Thus, we are not just looking whether ReDiscover or HBA predicted correct relationships, but rather we are considering whether or not they predicted the most informative ones. For example, because the *reordered-rows* relationship is a special case of the *equal* relationship, their features overlap. So, if ReDiscover and HBA predicted the *reordered-rows* relationship between

datasets A and B , then they would necessarily predict the *equal* relationship between them as well. However, the *reordered-rows* relationship is more informative than the *equal* relationship.

Part 3: Evaluating ReDiscover’s Prediction Score

We analyzed ReDiscover’s prediction scores from Part 1 in order to evaluate their accuracy, and to see whether there is a correlation between ReDiscover’s prediction score and the likelihood of a relationship. For each of the five test sets, we divided ReDiscover’s ranked lists of predictions into five bins based on the prediction score (100-80, 80-60, 60-40, 40-20, and 20-0). Then for each bin, we computed the percent of pairs with the relationship and the percent of pairs lacking the relationship. Finally, we computed the overall average prediction score of pairs that had the relationship, and those that did not have the relationship in each bin across the five test sets (see Figure 5.5).

5.3.2 Results

Figure 5.3 shows the average interpolated precision at standard recall levels (0.00, 0.10, . . . , 1.00) for ReDiscover and HBA predictions of *duplicate*, *row-containment*, *template*, *complementation*, and *reordered-rows* relationships on the five test datasets. The precision-recall curve is commonly used in evaluating ranked retrieval results for an information retrieval system. We interpolated the precision by using the maximum precision obtained at standard recall level i for any actual recall level greater than or equal to i [71]. We also labeled the precision-recall curves of both ReDiscover and HBA with their average prediction score across all tested relationships at each standard recall level (i.e., the average prediction score for all pairs up to a given standard recall point). In Figure 5.4, we show the interpolated precision per individual relationship.

As shown in Figure 5.3, ReDiscover performed better than the baseline, HBA,

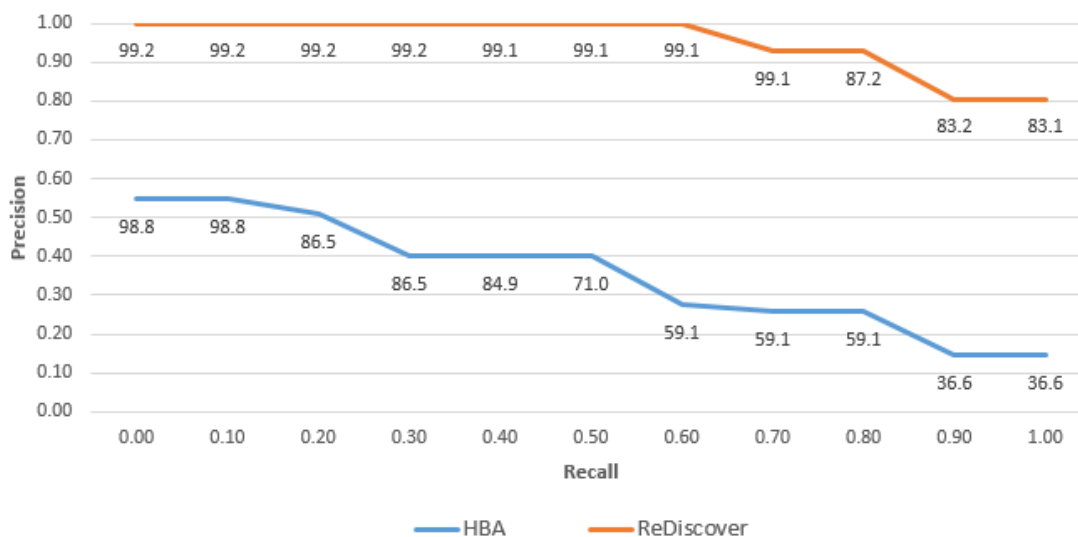
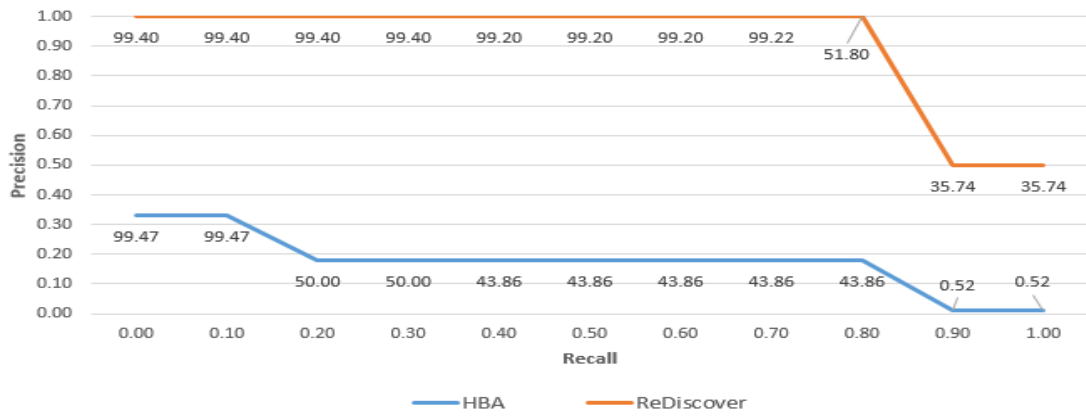


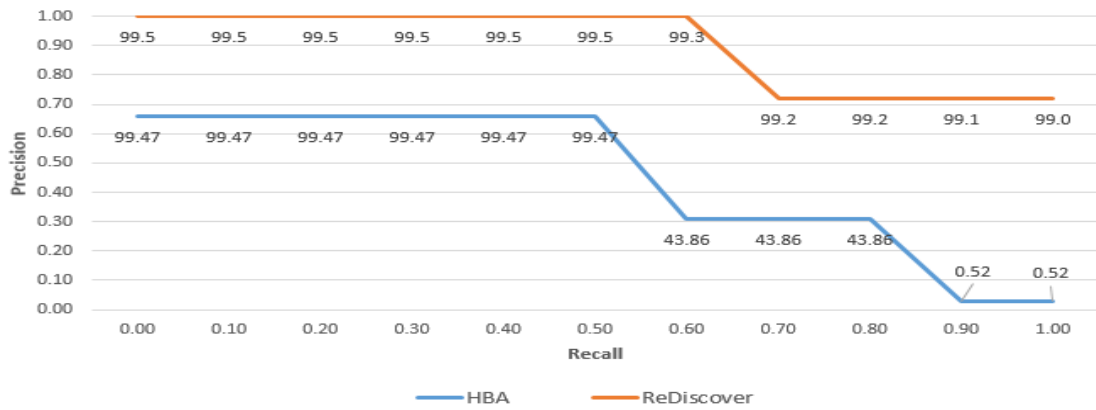
Figure 5.3: Average interpolated precision at standard recall levels, with average prediction scores across all relationships

at all standard recall levels for all tested relationships. ReDiscover started with a perfect precision of 1 where HBA started at 0.55 precision as its top predictions included incorrect relationships. The precision of the baseline started declining after recall level 0.2, where ReDiscover’s precision declined slightly only after recall level 0.6. ReDiscover maintained a precision rate of 0.8 till it predicted all existing relationships while the baseline precision dropped to 0.15 at recall level 0.9, which means that it ranked many incorrect predictions before all the correct ones.

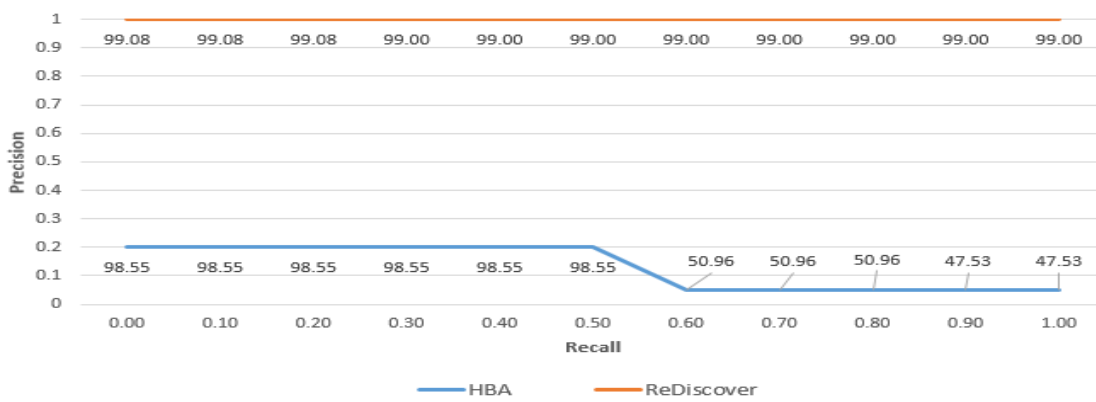
We can also conclude from Figure 5.4 that even when we break the interpolated precision down by individual relationships, we still see that ReDiscover outperformed HBA in each of the five tested relationships. In Particular, ReDiscover predicted the row-containment (c) and complementation (d) relationships with perfect precision, where HBA performed poorly in the row-containment relationship. For the duplicate (a), template (b) and reordered-rows (e) relationships, ReDiscover performance dropped after recall levels 0.8, 0.6 and 0.7 respectively.



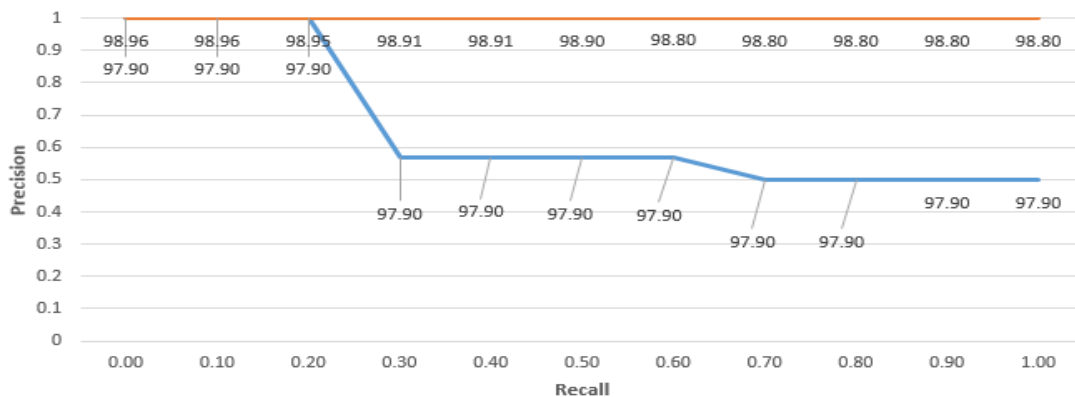
(a) Duplicate



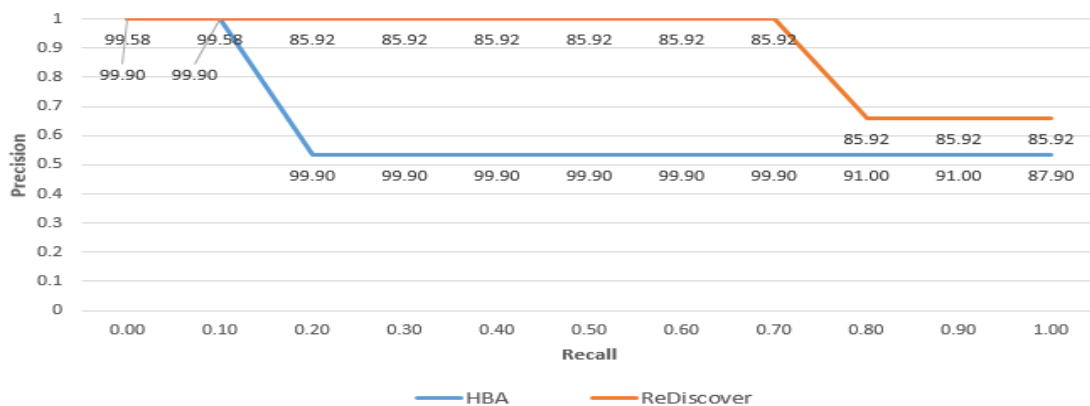
(b) Template



(c) Row-Containment



(d) Complementation



(e) Reordered-Rows

Figure 5.4: Interpolated precision at standard recall levels, with prediction scores.

When we analyzed the results we found that ReDiscover predictions were all correct, but some of the top-ranked predictions were less informative than the best predictions. We treat such predictions as false negatives (for computing precision). For example, when evaluating ReDiscover's performance on predicting the reordered-rows relationship, we found that it predicted the duplicate relationship

for some dataset pairs with the reordered-rows relationship with higher prediction scores than those of the reordered-rows predictions. As a result, the ranked list of predictions contained pairs with less-informative relationships before the more-informative ones.

On the other hand, HBA’s results for the five tested relationships contained incorrect predictions, hence, the poor performance of HBA compared to ReDiscover. However, HBA’s performance in predicting the complementation and reordered-rows relationships was better than that of the duplicate, template and row containment relationships. While some of the human-based features are indicative of the complementation and reordered-rows relationships, these features are not based on the content of the datasets but rather on their schema and metadata. In spite of that, and as shown in Figure 5.4, the precision-recall curve of HBA and ReDiscover never ended at the same point for any of the five tested relationship.

5.3.3 Discussion

Regarding the question of how far a user should go down the ranked list of predictions to find all correct relationships, Figure 5.3 shows that ReDiscover predicted all existing relationships between pairs of datasets in our test sets with over 0.7 precision. This high precision means that ReDiscover’s list of predictions up to the recall level of 1.0 contains very few incorrect predictions. Based on HBA’s results, a user can only find 80% of all existing relationships with 0.33 precision. Looking another way, a user would only have to examine 34 pairs to find all 24 relationships with ReDiscover, whereas with HBA he or she would need to look at 124 pairs to find just 6 of the related pairs.

To answer the question of how much user effort is wasted as a result of false positives, we computed the average non-interpolated precision of both ReDiscover and HBA in predicting the *duplicate*, *containment*, *template*, *complementation*, and

reordered-rows relationship between all pairs in our test datasets. The average non-interpolated⁴ precisions were 0.87 and 0.37 for ReDiscover and HBA, respectively. The average non-interpolated precision measures the performance over all relevant relationships, and it rewards systems that rank correct relationships highly [71]. We conclude from the results that ReDiscover generated a low rate of false positives, and users are less likely to waste time investigating unrelated datasets using ReDiscover’s predictions as compared to manual methods.



Figure 5.5: The results of ReDiscover’s prediction score evaluation (Part 3).

The results of part three of our evaluation, shown in Figure 5.5, suggest that ReDiscover’s prediction scores are highly correlated with the likelihood of a relationship (Q3). ReDiscover’s average prediction scores were between 80-100 for 92% of the pairs where the *duplicate*, *containment*, *template*, *complementation*, and *reordered-rows* relationships hold. For over 90% of the pairs lacking these relationships, ReDiscover’s average prediction scores were between 0-20. ReDiscover was

⁴“The measure is not an average of the precision at standard recall levels. Rather, it is the average of the precision value obtained after each relevant document is retrieved.” [71]

able to produce such good prediction scores because of the ability of its SVMs classifiers to distinguish between classes with high accuracy, which is a result of the use of distinctive relationship features and comprehensive training datasets. However, our evaluation was on a subset of the relationship types and we might not do as well on other types. Additionally, there are generally a lot more pairs without a relationship than that with the relationship—5% of pairs without a relationship is likely larger than 5% of pairs with a relationship. We can also conclude from the results that our cut-off threshold of 30 for reporting results is quite reasonable—a user would encounter very few false positives, while missing less than 3% of the actual relationships.

In conclusion, the results of our evaluation shows that ReDiscover is a viable approach for predicting relationships between scientific datasets in a collection. ReDiscover achieved such performance because of its ability to compute distinctive relationship features based on column summaries, column correspondences and spreadsheet metadata. Furthermore, using Bloom filters to compute fast approximation of categorical columns enabled ReDiscover to extract powerful features, such as *isAllBVoFAContainedInB*, *avgDiceSimilarityforDuplicate* and *avgDiceSimilarityforContainment*. In the next chapter, we discuss research work related to scientific data management (Section 6.1) and to the ReConnect (Section 6.2) and ReDiscover (Section 6.3) systems.

CHAPTER 6: RELATED WORK

To the best of our knowledge, we are the first to provide a methodology for helping scientists determine connections between datasets in a collection in the absence of explicit provenance or history. In this chapter, we provide a review of tools and techniques, drawn from several research areas, related to our work. In the first section, we address systems that scientists may use to manage collections of datasets, and discuss why these systems are not adequate for determining which dataset to select for a given task. In the second section, we address tools and techniques that are similar or relevant to our relationship-testing approach (ReConnect). In the last section, we review work related to the relationship-prediction methodology; specifically we review data extraction techniques, schema matching using machine learning, and summarizing categorical data using Bloom filters.

6.1 SCIENTIFIC DATA MANAGEMENT

Managing file-based datasets with database systems. Scientific data management systems, such as SciDB [21] and the Scientific Data Service framework [31], can help scientists manage their research data. However, as is the case with DBMSs, such systems require users to have a good technical background to be able to do so. Furthermore, because these systems are based on multidimensional arrays, the scientist has to convert his or her file-based datasets, such as spreadsheets, into the multidimensional-array format. Thus, the scientist still needs help with determining which dataset to convert to the formats that are compatible with these systems.

Other scientific data management systems attempt to manage file-based datasets by attaching them to a database system. For example, Data Vaults, a scientific data warehouse developed by Ivanova et al. [44], allows scientists to attach an external file repository to the DBMS, and then access the data and metadata of these files using a query language. However, Data Vaults assumes that files contain raw data in the form of CSV or standard scientific file formats, such as MSEED or GeoTIFF, and is not equipped with data-extraction capabilities for files that may contain semi-structured datasets, such as spreadsheets.

Alagiannis et al. [4] also recognize that an impediment for using database systems in scientific analysis applications is the complexity of loading data into a database and the data-to-query time—the initialization cost of loading data and preparing the database for queries. Their approach to overcoming such an impediment is to fully integrate “row data” files in the database query engine. To that end, they developed PostgresRaw, a database system that provides incremental files data loading, on-the-fly indexing and caching to support faster future queries and improve query performance. However, their approach requires that the schema of a dataset must be known a priori, and—as is the case with Data Vaults—it does not provide data extraction capabilities for files that may contain semi-structured datasets. Furthermore, for a scientist to use PostgresRAW, he or she must know how to write SQL queries, and understand how datasets are connected. Our relationship-identification methodology automatically extracts datasets from spreadsheets, and helps researchers decide how to work with data stored in their file-based datasets by determining connections between them.

Several approaches [3, 8, 15, 19, 22–24] suggest converting spreadsheets into a relational model to enable managing their data using relational databases and data-integration tools. However, none of these approaches addresses the problem of identifying which datasets of interest should be converted from a collection of datasets. Some of these approaches can also lose valuable information that can be

used in predicting relationships between datasets. For instance, Cunha et al. [24] developed a system that attempts to fix spreadsheet errors by extracting their true (normalized) relational schema. In the process of converting a spreadsheet's data into a relational schema, Cunha's approach loses the original spreadsheet schema. Our relationship-identification methodology needs to preserve the original schema in order to identify the original connections between spreadsheet data. Additionally, Cunha et al. convert spreadsheet tables into sets of rows (relations) in which the order of rows is not taken into account. Our methodology captures dataset order and uses it in predicting ordered relationships such as prefix and subsequence.

Version control systems (VCS). Ram [62] showed that VCSs, which are used in the software industry to maintain software code repositories, can be leveraged in managing scientific data such as datasets, experiment notes, and manuscripts. The use of VCS tools, such as Github [37] and SourceForge [67], in science can facilitate collaborations and enhance scientific data reproducibility.

Schopf [66] also proposes that data should be managed in a way similar to how production software is managed. To better manage data and produce quality data, Schopf argues that data should be treated as ongoing process. Such a process considers that data are manipulated by several contributors; may go through cyclical releases that include bug fixes, derivation history tracking and versioning; and producing licensing and citation information with each version release.

While VCSs are very effective tools for keeping track of the history of modifications to files over time, which might reveal something about the connection between a dataset and one of its earlier versions, that information does not necessarily help with determining the relationship between versions on different branches or between independent datasets.

Data provenance and scientific workflow management systems (SWMS).

SWMSs enable scientists to organize and execute a series of computational steps (i.e., workflows) on data collected from several sources. Many of these systems, including Kepler [7], SciCumulus [27], Chimera [35], VisTrails [13] and MyGrid [69], are equipped with provenance-tracking functionality, which records important provenance information to help scientists document the lineage evolution of their data and the processes used to manipulate it.

Some SWMS systems track the provenance of both data objects and workflows. For example, the Kepler scientific workflow system [7] collects provenance information about the standard data lineage (i.e., derivation history), and the context in which the workflow was executed. Other SWMS systems focus on collecting provenance information about data objects. For instance, the Chimera virtual data system [35] tracks the derivation path of a data product (i.e., dataset) to help a scientist reproduce a derived data product, and to validate the results of an experiment. Furthermore, there are systems that provide various capabilities for tracking the evolution of a workflow from one version to the other. An example of such systems is the VisTrails, which [13] provides several capabilities. First, it allows users to explore variations in the design history of a workflow. Second, it helps a user to determine whether two different workflows share any common elements or if they were derived from the same root.

While provenance-tracking systems can aid scientists in uncovering relationships (or connections) between their datasets, they cannot track provenance of data processing outside of the provenance tracking system (offline processing). Davidson et al. [26] stated that “When such analyses [of intermediate workflow results] are carried out by hand or automated using general-purpose scripting languages, the means by which results are produced are typically not recorded automatically,

and often not even recorded manually.” Consequently, determining connections between such intermediate results can help scientists bridge such gaps in the derivation history of their datasets. Our relationship-identification approach can detect informative relationships that can help provenance-tracking systems infer offline data-transformation activities.

6.2 RELATIONSHIP TESTING

We review work related to our relationship-testing methodology below. First, in Section 6.2.1 we review change-inference tools that may be used for determining relationships between two spreadsheets. We also address Record Linkage techniques and discuss how they might be used to improve ReConnect. Then, in Section 6.2.2 we examine techniques that have influenced the way we designed ReConnect, such as data profiling, and the Clio and Bellman systems.

6.2.1 Similar tools

Change-inference tools [16, 34, 56, 75] may help users understand simple relationships between small spreadsheet instances. By using these tools, users may infer relationships through analyzing the change lists they generate. As our evaluation results show (Section 3.2.2), none of these tools can provide adequate help for users in identifying simple relationships in spreadsheets with hundreds of rows or columns. Furthermore, in the case of two spreadsheets that contain the same data but have different row or column orders, current change-inference tools do not detect that they have identical data.

The problem of Record Linkage (RL)—identifying records coming from different sources and representing the same real world entity—has received significant attention from statistics and computer-science researchers [32]. We believe that

some RL techniques may be useful to extend our work in two ways. First, field-matching techniques, including character-based techniques such as edit distance, or token-based techniques such as Q-Grams with tf-idf [39], can be used to improve our column-correspondence process by enabling ReConnect to better match similar column headers. Second, we may be able to use record-matching techniques, such as Automated Object Matching [76], to realize additional relationships (e.g., a near-match relationship suggested by one of our user-study participants). However, we still need to do a higher-level analysis of the results of RL techniques in order to detect such relationships. Giving users back a matching list of identical records still puts the burden upon them of analyzing the list to understand how the datasets as a whole connect. As we showed in evaluating change-inference tools (Section 3.2.2), ReConnect is working on an abstract level, allowing scientists to understand connections among their datasets without having to analyze individual records.

6.2.2 Relevant Techniques

ReConnect adapts Clio’s [41] idea of integrating users’ feedback in the schema-mapping process. Clio, a semi-automated tool that maps column names of two database tables, has an interactive user interface that allows users to dynamically provide feedback on proposed schema mappings. As a result, users have full control over column mappings and can map similar schemata differently for various purposes. After computing an initial column correspondence, ReConnect allows users to correct the computed correspondence, which improves the accuracy of the schema correspondence process and allows users to test various correspondences. ReConnect also provides the *explore sub-correspondence* feature, which automatically searches within the current correspondence for the sub-correspondence that produces the largest set of common rows between spreadsheet pairs and reports that correspondence to the user.

Data profiling, the process of gathering and examining statistical summaries of data to understand its structure and content, is commonly used in data cleaning and data integration [10,61]. Data-cleaning applications use profiling information to aid in analyzing different aspects of attributes' quality. For example, the max and min can be used to check whether or not the values of a given attribute (data column) are within the expected range. In our work, we use data profiling differently: as an aid in identifying attribute correspondences between two different schemas, which is similar to the way it is used in data-integration applications [30].

The Bellman system [25] is a browser for complex databases that provides tools and services to help users discover the structure of databases. ReConnect collects the same statistics that the Bellman system collects, including the number of rows, the number of distinct values in a column, the number of null values per column, and the ten most common values in a column along with their respective frequencies. However, the Bellman system uses this profiling information to help data analysts understand the structure of a database, whereas ReConnect uses it to help users connect columns correctly and to guide the process of relationship identification.

6.3 RELATIONSHIP PREDICTION

The relationship-prediction methodology in ReDiscover builds upon a number of techniques from several research areas, such as relational databases and machine learning. In this section, we review techniques relevant to structured data extraction (Section 6.3.1), automatic schema-matching techniques (Section 6.3.2), and categorical data summarization using Bloom filters (Section 6.3.3).

6.3.1 Tabular Dataset Extraction

Our relationship-prediction methodology extracts tabular datasets from spreadsheets and analyzes their data to recover information that can help a user in identifying the best dataset(s) to work with, or how to work with data stored in their datasets (e.g, combining complementary datasets or removing redundant ones). The work on recovering semantics of tables on the web by Ventetis et al. [72] is the work most related to ours in this respect. They developed an approach for automatically recovering semantics of tables on the web by 1) adding annotations to the columns of a table, and 2) using these annotations to determine binary relationships represented by the columns in that table. More specifically, their approach labels a column A with class C (e.g, species, city) if the majority of the values in A are labeled with class C in the *isA* database—a database that contains a set of pairs of the form (instance, class). Then, it labels the relationship between a pair of column (A, B) in a table with relationship R (e.g., *is Known as, capital of*) if the majority of pairs of values from A and B occurs in the *relations* database—a database of triples of the form (argument1, predicate, argument2). Then, their approach uses column labels and relationship labels to support table information retrieval, ranking and other operations, such as combining related tables via joins or unions.

Our approach is different for the following reasons. First, we use data-profiling and Bloom filter techniques to summarize column values, and then we analyze column summaries to determine column correspondences and to predict relationships between datasets. On the other hand, Ventetis et al.’s approach examines all the values of a column to determine their semantic labels. It also examines all pairs of values between two columns to identify their relation label. Second, their techniques use information extracted from the web, including the *isA* and *relations* databases, to associate semantic information to structured tables on the web, and

then use these recovered semantics to determine related tables. Our methodology determines relationships between datasets by extracting indicative features from the data stored in the datasets without relying on external sources. Lastly, our approach targets scientific datasets stored in spreadsheets, while Ventetis et al. target generic tables on the web. Nevertheless, we believe that we can use recovered semantic information from tabular datasets to compute indicative relationship features that could improve the performance of our relationship-prediction approach.

A number of researchers have proposed approaches for extracting data from spreadsheets [3, 8, 15, 19, 22–24]. For example, the FlashRelate system [8] allows spreadsheet users with no programming experience to convert ad-hoc data structures into relational ones. However, FlashRelate is not suitable for extracting data from a large collection of spreadsheets because users need to provide positive and negative examples of the desired relational structure from each spreadsheet.

Abraham et al. [3] also developed a system, called UCheck, for extracting tables from spreadsheets. UCheck is based on a unit-reasoning technique that exploits label and header information in spreadsheets to identify tables. This unit-reasoning technique examines the extracted header information to validate the consistency of cell data and formulas, which allows users to identify potential errors in their spreadsheets.

Cunha et al. [22] developed ClassSheets, a tool that applies relational database techniques to convert spreadsheets into the relational model, and it works as follows. First, ClassSheets detects all functional dependencies among spreadsheet columns. Next, it attempt to filter as many accidental functional dependencies as possible. Then, it uses the resulting functional dependencies to identify the relational schema with candidate primary and foreign keys. After that, it generates and refactors the relational intermediate directed (RID) graph using the resulting relational schema. RID is a graph data structure that represents the relationships

between schemas in a relational model. The graph nodes represent schemas and directed edges represent foreign keys between these schemas [22]. Lastly, ClassSheet translates the relational graph into a ClassSheet (table). The goal of their work is to automate the generation of refactored spreadsheets from the inferred ClassSheet model. However, as is the case with UCheck, the Class Sheet work focuses on helping users avoid spreadsheet errors. Further, by converting spreadsheets into the relational model, ClassSheet might discard important information, such as row order and the original spreadsheet schema, which ReDiscover uses in predicting relationships.

ReDiscover adapts SENBAZURU’s [19] idea of using CRFs in extracting spreadsheet data. SENBAZURU extracts relational data from spreadsheets and offers several relational operations over the extracted data. However, there are two fundamental differences between ReDiscover and SENBAZURU. ReDiscover operates on finer level of granularity (spreadsheet cells) than SENBAZURU, which operates on spreadsheet rows. As a result, ReDiscover can detect data columns that are stacked vertically or horizontally, whereas SENBAZURU assumes that data frames can only stack vertically. Second, SENBAZURU is designed with the goal of inferring hierarchical structures from *data-presentation* spreadsheets, such as spreadsheet reports downloaded from the web (e.g, U.S. Census Bureau reports). These spreadsheets often contain processed data developed when organizing data for human consumption. Figure 6.1 shows an example of a data-presentation spreadsheet, which contains information about the United States population between 1950 and 2009 [14]. The first column of this spreadsheet represents four different demographics: sex, race, Hispanic origin and year. The goal of such data formatting is to help humans understand the presented information in a compact and easy to understand way. Consequently, data-presentation spreadsheets often have derived fields (e.g., sums, averages), and they are often organized as hierarchical cross-tabs, which is not common for raw data. In our work, we focus

Resident population, by age, sex, race, and Hispanic origin: United States, selected years 1950-2009						
(Data are based on the decennial census updated with data from multiple sources)						
Sex, race, Hispanic origin, and year	Total resident population	Under 1 year	1-4 years	5-14 years	15-24 years	25-34 years
All persons	Number in thousands					
1950.....	150,697	3,147	13,017	24,319	22,098	23,759
1960.....	179,323	4,112	16,209	35,465	24,020	22,818
1970.....	203,212	3,485	13,669	40,746	35,441	24,907
1980.....	226,546	3,534	12,815	34,942	42,487	37,082
1990.....	248,710	3,946	14,812	35,095	37,013	43,161
2000.....	281,422	3,806	15,370	41,078	39,184	39,892
2006.....	299,398	4,130	16,287	40,337	42,435	40,416
2007.....	301,621	4,257	16,467	40,164	42,506	40,591
2008.....	304,060	4,313	16,693	40,120	42,573	40,932
2009.....	307,007	4,261	17,038	40,583	43,077	41,566
Male						
1950.....	74,833	1,602	6,634	12,375	10,918	11,597
1960.....	88,331	2,090	8,240	18,029	11,906	11,179
1970.....	98,912	1,778	6,968	20,759	17,551	12,217
1980.....	110,053	1,806	6,556	17,855	21,419	18,382
1990.....	121,239	2,018	7,581	17,971	18,915	21,564
2000.....	138,054	1,949	7,862	21,043	20,079	20,121
2006.....	147,512	2,113	8,329	20,640	21,845	20,565
2007.....	148,659	2,179	8,424	20,549	21,860	20,683
2008.....	149,925	2,208	8,540	20,522	21,873	20,900
2009.....	151,449	2,179	8,708	20,758	22,145	21,224
Female						
1950.....	75,864	1,545	6,383	11,944	11,181	12,162
1960.....	90,992	2,022	7,969	17,437	12,114	11,639
1970.....	104,300	1,707	6,701	19,986	17,890	12,690
1980.....	116,493	1,727	6,259	17,087	21,068	18,700
1990.....	127,471	1,928	7,231	17,124	18,098	21,596
2000.....	143,368	1,857	7,508	20,034	19,105	19,771
2006.....	151,886	2,017	7,959	19,697	20,590	19,851

Figure 6.1: An example of data-presentation spreadsheet: Resident population, by age, sex, race, and Hispanic origin: United States, selected years 1950-2009 (A partial picture of the original spreadsheet). Source: [14].

on *data-collection* spreadsheets that scientists commonly use for collecting tabular data (often as raw data).

Conditional random fields were also used for extracting tables from text documents. Pinto et al. [59] developed a CRF model that uses text content (e.g., alphabet characters, digit characters) and layout (e.g., white-space gaps, separator characters: +, -, :, !, =, *) as features for labeling each line of a text document with one of the following tags.

- *Non-extraction* labels for text lines that are not part of a table.
- *Header* labels for lines that contain a table’s column names.
- *Data row* labels for lines that contain data cells.
- *Caption* labels for lines that appear below data but apply to the table.

Pinto et al.’s approach is different from ours because 1) as is the case with SENBAZURU, it assumes that data tables can only stack vertically, 2) our approach extracts data columns from spreadsheets, whereas Pinto et al.’s approach extracts tables from text files, and 3) ReDiscover’s Label Cells process uses a more comprehensive set of features, including layout, text, content and context features, than those used in Pinto et al.’s approach.

6.3.2 Automated Schema Matching Techniques

Rahm et al. [60] presented a taxonomy that covers several automated schema-matching approaches, and classified them based on various criteria. Figure 6.2 shows their classification. We also highlighted in blue the schema-matching categories that applies to our approach. We discuss Rahm et al.’s classification criteria below.

- Individual versus combinational matchers: individual matching techniques use a single schema-matching algorithm, whereas combinational techniques are either *hybrid* matchers, which match schemas based on multiple criteria, or *composite* matchers, which use multiple independent schema-matching algorithms to match two schemas, and then combine the results of all algorithms to identify the best matching. ReDiscover’s column-matching approach is classified as an individual-matcher technique, as it only use a single matching algorithm.

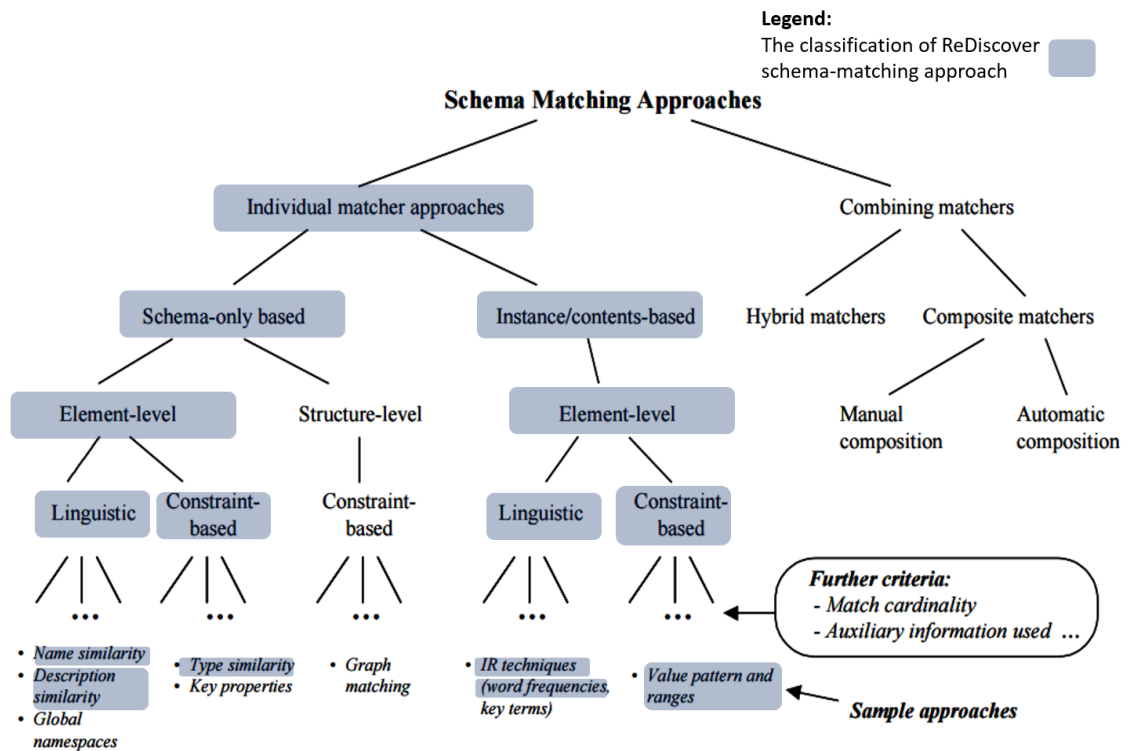


Figure 6.2: The classification of ReDiscover’s schema-matching approach based on Rahm et al.’s taxonomy (Highlighting was added). Source: [60].

- Schema- versus instance/content-based: schema-based techniques consider only schema information, such as column names, types and constraints, whereas instance-based techniques use attribute values or statistics derived from them to determine correspondence between schema attributes. Since our column-matching approach uses column names and column statistics, it can be classified as both schema-based as well as instance-based approach. However, ReDiscover compares column summaries of a pair of columns, and not their individual data values.

- Structure- versus element-level matching: structural schema-matching techniques compare groups of schema elements, such as tables, while element-matching techniques identify pairwise correspondence between individual elements, such as attributes. Our schema-matching approach can be categorized under element-matching techniques, as it only matches individual data columns.
- Linguistic-based: schema-matching techniques under this category use names of schema elements (e.g., column names) and other text descriptions to determine the correspondence between two schemas. Our approach is classified as linguistic-based because the SVM model of the Match Columns process uses column-name similarity in determining corresponding columns.
- Constraint-based: such approaches use schema constraints, such as data types, value ranges, value uniqueness, cardinality and referential integrity between schemas. ReDiscover’s column-matching approach can be classified as constraint-based approach, as it uses information such as min and max values, row count, unique and nulls values in matching a pair of data columns. However, because we extract tables from spreadsheets, there is no referential integrity information that we can use.
- Matching cardinality: does a schema-matching technique produce one-to-one mappings? Or does it produce one-to-many (or many-to-one) mappings? The matching cardinality of ReDiscover’s column-matching approach is one-to-one.

Several *instance-level* matching approaches [29, 30, 51–53] use machine-learning techniques, such as neural networks and traditional classification methods, as follows. First, value instances of the first schema are characterized and matched, on a one-to-one basis, with value instances from the second schema. Then, the resulting per-instance match results are merged and abstracted to the schema level.

Our approach is different because we do not match columns based on one-to-one instance matching between column values. Instead, we match columns based on similarity vectors, which summarize all the instances in a data column.

The instance-level approach most closely related to ours is the one used in SEMINT system [53]. SEMINT matches attributes between two database schemas by first computing *matching signatures* derived from schema information (e.g., attribute name, data type, field length, primary key, foreign key), and data contents and statistics (e.g., minimum, maximum, average, standard deviation). Then, the SEMINT system clusters attributes based on the Euclidean distance between their matching signatures. While SEMINT and ReDiscover both use attribute (column) statistics for matching, they are different in several aspects. First, the SEMINT system identifies corresponding attributes in different DBMSs that represent the same real-world information, hence it works on well-defined tables, and utilizes schema information such as primary and foreign keys. In contrast ReDiscover matches the columns of semi-structured tabular datasets stored in spreadsheets. Second, SEMINT statistics are computed from a small sample of an attribute’s row values, whereas ReDiscover computes statistics from all the row values of a column. Lastly, for matching character data, SEMINT computes statistics on the number of bytes used to store that data. ReDiscover uses Bloom filters to approximate similarity between bit vector representation of categorical data.

Most of the work on schema matching has focused on a particular data model (e.g., databases, XML documents) or application (e.g., data integration, data warehousing). Thus, the design decisions of such schema-matching techniques are significantly influenced by the application domain. For instance, schema-matching techniques for data-integration applications are designed to determine similar schema structures from a set of well-defined tables that often model similar real-world concepts. In contrast, ReDiscover’s schema-matching approach is designed with the goal of determining column correspondences between ill-structured tabular

datasets that model a variety of concepts. Further, we are working in situation where some datasets are unrelated, and we do not know that a priori.

Techniques used in foreign-key discovery, which are aimed at detecting semantic association between primary- and foreign-key attributes (columns) in relational databases, are also relevant to ReDiscover’s column-matching technique. Rostin et al. [64] presented a machine-learning approach that computes all inclusion dependencies (IND) between attributes to find candidate foreign keys. Then, it uses a binary classification algorithm to determine the true INDs, and hence the true foreign-key attribute pairs. While ReDiscover and Rostin’s approach share some features for detecting column association (matching), such as column-name similarity, value-range inclusion and table-size ratio (the ratio of the number of rows in two columns), Rostin’s features are engineered for detecting foreign-key associations. Consequently, these features are not adequate for general semantic matching between columns. On the other hand, ReDiscover uses a set of features that measure similarity between columns statistics and Bloom filters of two columns, and hence are more suitable for detecting general column correspondence, as we discussed in Section 4.2.4.

6.3.3 Similarity Detection using Bloom Filter

In this section we discuss related techniques that use Bloom filters for set (or string) similarity detection. Jain et al. [45] developed a technique for detecting duplicate (or near duplicate) documents in the results of a search engine. They use Bloom filters to detect similar documents by first using Content-Defined Chunks (CDC) to extract modification-resilient document features. Then, they use these features to compute a Bloom filter for each document. Next, to detect whether two documents are near-duplicate, they compare the Bloom filter of one document with that of another by computing the bit-wise AND between them. As we discussed in Section 4.2.4, ReDiscover also uses Bloom filters for computing similarity (between

two columns). However, we use the estimated Dice similarity coefficient for measuring similarity between two Bloom filters. Further, ReDiscover uses Bloom filters for summarizing column content, where as Jain’s approach uses it for summarizing web-document content.

Schnell et al. [65] proposed a method for linking the records of multiple databases with additional information about the same person (patient). Because in the medical field it is very important to maintain the confidentiality of patients records, they use Bloom filter for encrypting patient information (e.g., patient name) as follows. First, they split each record identifier into sets of consecutive bigrams (2-grams). Then, they compute a Bloom filter for all the bigrams of an identifier. Lastly, they use the Dice coefficient to compute similarity between the Bloom filters of the record identifiers they are trying to link. While Schnell’s approach and ReDiscover share the use of the Dice coefficient for computing similarity between two Bloom filters, the Bloom filters in the two methods represent different objects. In ReDiscover, Bloom filters represent the content of a data column, whereas in Schnell’s approach they represent bigrams of a person’s record identifier. Additionally, the two approaches use Bloom filters for different purposes. In ReDiscover, it is used to summarize data columns to avoid extensively analyzing individual column values when computing similarity between columns, while in Schnell’s approach, it is used mainly for preserving the confidentiality of patient-record identifiers. Further, we use n-grams as a way of encoding some order information into Bloom filters.

CHAPTER 7: FUTURE WORK, FURTHER APPLICATIONS, AND CONCLUSIONS

In this study we explored the problem of determining relationships between scientific datasets in a collection of spreadsheets, introduced a relationship-identification methodology as a solution, and developed two prototype systems to assess our methodology. We first developed ReConnect, a tool for identifying relationships between two datasets. Encouraged by its methodological evaluation and the promising results from our user study, we then extended our methodology to predict relationships in a collection of datasets. We developed ReDiscover, an end-to-end prototype system that predicts, from a collection of datasets, related pairs and their possible relationship. The preliminary evaluation of ReDiscover shows promising performance and areas for further investigation.

Informed by the feedback we received from our user study subjects, our research collaborators, observations we made from analyzing the datasets we gathered and the results of our evaluation, we have identified a number of new research directions. We also recognized several other application domains for our methodology. In this chapter, we discuss future extensions of our work (Section 7.1), explore other application domains for our relationship-prediction methodology (Section 7.2), and lastly conclude this dissertation in Section 7.3.

7.1 FUTURE WORK

This work focused on determining pairwise connections between scientific datasets. The first direction we want to explore in the future is extending our methodology

to detect multi-dataset connections (Section 7.1.1). The second direction for exploration is scaling to enterprise-level collections of datasets (Sections 7.1.2). The third direction is enhancing the relationship-prediction performance of ReDiscover (Section 7.1.3). The last direction we discuss is integrating ReConnect with ReDiscover to provide users with the ability to predict and validate relationships with a single system (Section 7.1.4).

7.1.1 Multi-Dataset Connection Identification

As individual relationships are discovered, we can use that information to make further inference about multi-dataset connections. A connection is identified based on a certain combination of pairwise relationships. For example, if dataset A contains B , and C duplicates B , then A contains C as well.

Another future direction would be the identification of the *concatenation* connection among multiple datasets in a collection. If spreadsheet Y was formed as the concatenation of spreadsheets $\{X_1, \dots, X_n\}$, or a subset of these spreadsheets, then the following relationships exist:

- a. some X_i is a *prefix* of Y , where $0 < i \leq n$,
- b. some X_j is a *suffix* of Y , where $0 < j \leq n$,
- c. zero or more X_k are *infixes* of Y , where $1 < k < n$, and
- d. all X_l are pairwise disjoint and $\cup X_l = Y$, where $1 \leq l < m$ (where m is the total number of spreadsheets involved in the concatenation).

We are also considering a graph of the relationships discovered so far in a collection to show to the user, such as the one shown in Figure 7.1. Such a graph would show related datasets and could include several sub-graphs, each of which represents a group of related datasets within a collection. The advantage of the graph feature is that it would provide the user an overview of his or her

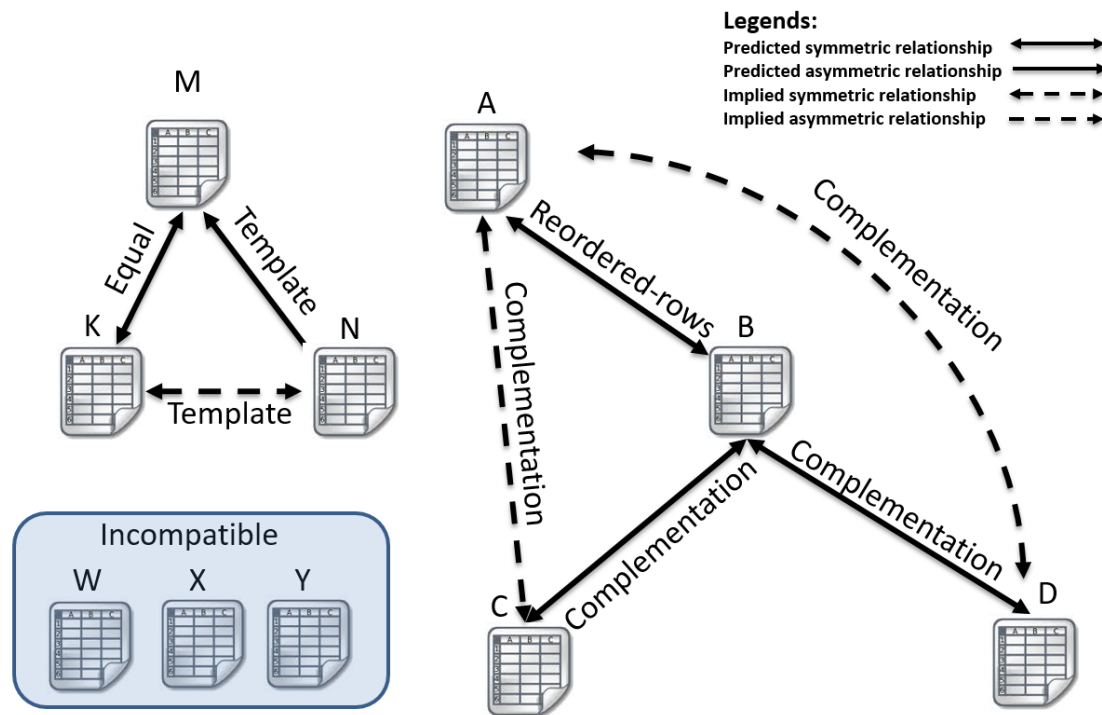


Figure 7.1: An example of a graph of predicted relationships in a collection.

collection and help her determine which pairs to select for analysis, publishing or sharing efficiently. The graph feature is especially important when analyzing very large (enterprise-level) collections of datasets. We discuss the challenges and opportunities for scaling to larger dataset collections in the next section.

7.1.2 Scaling to Large Dataset Collections

In this work, we focused on the feasibility of automating our relationship-prediction methodology, and we targeted collections arising from individual scientists and small groups of researchers. In the future, we want to scale our methodology to larger (possibly enterprise-level) collections of datasets. As we discussed in Section 4.2, ReDiscover’s architecture is designed to scale, as it avoids extensive manipulation of individual values and only uses bounded-size feature summaries for comparison on $O(n^2)$ tasks. The main consideration in scaling for larger dataset

collections is the ability to parallelize our system. The first three processes of ReDiscover (Label Cells, Extract Columns and Compute Column Summaries) operate on a per-dataset basis, thus we can easily run these processes for multiple datasets in parallel.

The second part of ReDiscover (Match Columns and Predict Relationships) operates on a per-dataset-pair basis. For this part, we have tried to bound the computation time on pairwise operations by working with column summaries. As pairs can be considered independently, parallelism can probably help us scale to thousands of datasets, exhaustively comparing millions of pairs. After that, we need a way of avoiding the n^2 comparisons.

The first possible approach that we plan to investigate to avoiding the n^2 problem is based on blocking techniques from Entity Resolution (ER)—the task of matching and linking different instances of the same real-world entity [36]. ER techniques also struggle with the n^2 problem as they need to consider all possible entity pairs in a large collections. *Blocking* reduces the number of entity comparisons by grouping similar entities and comparing only entities in the same group. (Groups can overlap.) In particular, *meta-blocking* techniques, such as the supervised meta-blocking technique [58], looks promising for our application. This technique applies supervised learning algorithms to develop classification models for quickly distinguishing between redundant or superfluous comparisons and promising comparisons. Such a technique can significantly reduce the number of comparisons and hence allows for scaling to large dataset collections.

Figure 7.2 shows one way of applying blocking techniques to scale our approach. First, ReDiscover computes a Bloom bit-vector for each data-column in a dataset. Then, it clusters similar columns together based on their bit-vectors. Next, ReDiscover uses the resulting column clusters to group related datasets, which are the datasets with columns that share the same cluster. More specifically, each group of columns (C) induces a group of datasets (D). That is, $D(C) = \{d \in D \mid \exists c \in C$

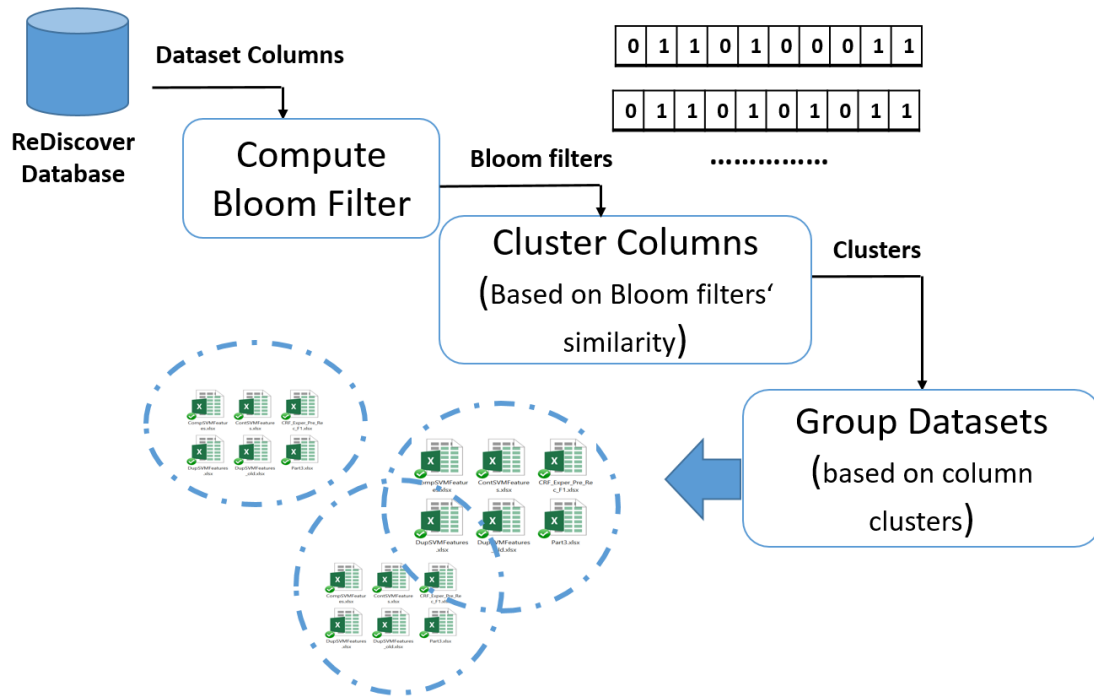


Figure 7.2: A possible approach for using blocking techniques to scale ReDiscover.

and c is a column of d). Now that related datasets are separated into smaller groups, ReDiscover can distribute these groups to several nodes and process the last two stages, Match Columns and Predict Relationships, for the resulting groups simultaneously.

Another way of applying blocking would be by adapting an Information Retrieval (IR) approach, namely *document retrieval*, to deal with the n^2 problem. Our approach would treat every dataset in the collection as a document, and its column values as terms. Then, the approach builds a full index for all documents in a collection. Next, using the index, ReDiscover computes the top- k matching datasets for each dataset in the collection. Because it only considers a dataset and the top- k best-matching datasets for potential pairs, ReDiscover would reduce the number of datasets to process for relationship prediction. Document search itself can be parallelized by term or document [57]. ReDiscover could also create an index

of terms and the documents in which they occur to cluster related datasets. Such an index would help ReDiscover group documents based on the number of shared terms, and process the last two stages of the relationship-prediction approach for several clusters in parallel. However, these methods are based on heuristics and might miss related pairs.

7.1.3 Improving ReDiscover’s Prediction Performance

We discuss here ways of further improving ReDiscover’s prediction performance, including evaluating relationship-prediction features, conducting further user studies, improving column matching, and exploiting other spreadsheet features.

Evaluating Relationship-Prediction Features

While the results of ReDiscover’s preliminary evaluations showed promising relationship-prediction performance, we believe that there is still room for improving its accuracy and efficiency. However, there will be always a trade-off between accuracy and efficiency: fewer features means faster classification time, but less classification accuracy. We can identify the best balance between accuracy and efficiency by examining the relative performance of various combinations of relationship-prediction features.

In the future, we want to experiment with several combinations of existing and new relationship-prediction features, such as features based on spreadsheet formulas. For each combination (feature set), first we will need to use it to train a new SVM model. Next, we will use the new classification model to examine its relationship-prediction accuracy and its classification time, which is the average time to classify a pair of datasets. Lastly, we will select the feature set that produces the best balance between prediction accuracy and classification time.

Conducting Further User Studies

In this work, we compared ReDiscover to an approximated human baseline approach (Section 5.3.1), to see whether its predictions were better than strategies that a human might use to identify relationships between pairs of datasets in a collection. In the future, we would like to conduct additional user studies to 1) evaluate the usefulness of ReDiscover in simplifying the task of determining related pairs of datasets in a collection, 2) receive feedback from the study participants about their experience with ReDiscover, and 3) receive feedback about ways of improving ReDiscover functionality and user experience.

Improving Column Matching

Exploiting user feedback. The results of ReDiscover’s preliminary evaluation (Section 5.1) showed that cell labeling and column matching are important for accurate relationship predictions. However, semi-structured data extraction and schema matching are hard problems. Even limited feedback from users can significantly improve these tasks. Currently, ReDiscover allows users to fix cell-labeling errors and retrain the labeling algorithm to avoid similar labeling errors in its future predictions. We plan to also exploit users’ feedback in column matching by implementing a visual column-correspondence user interface (UI). This UI will enable users to quickly repair column mappings. User repairs can be used to retrain our Match Columns classifier as well.

Handling ambiguous column matching. In Section 4.2.4, we discussed column-correspondence ambiguity issues and introduced the Enhance Column Correspondence method (Algorithm 3), which removes ambiguity by ensuring that the resulting column correspondence consists of only one-to-one mappings. However, when reducing a one-to-many mapping, ReDiscover favors the mapping with the highest score without considering the effect of selecting such a mapping on the quality

of the overall resulting column correspondence. Thus, the Enhance Column Correspondence algorithm is characterized as a greedy method that makes locally optimum choices without guarantying global optimality. We want to enhance this algorithm by generating several enhanced column-correspondence alternatives and choosing the one that produces the best overall column-correspondence score.

Improving column matching for datasets with limited-domain columns.

Our current column-matching approach does not perform well with datasets that consists of several limited-domain columns—columns with few domain values. For example, assume that dataset *A* has *Animal Category* and *Animal Category2* columns and dataset *B* has *Animal Class* and *Animal Class2* columns. The *Animal Category* and *Animal Category2* columns of *A* closely resemble both *Animal Class* and *Animal Class2* of *B*. Consequently, ReDiscover would match column *Animal Category* to columns *Animal Class* and *Animal Class2*, and *Animal Category2* column to *Animal Class* and *Animal Class2*, as the summaries of these columns are very similar. Using value-mapping (instance-based) matching techniques can aid in removing ambiguous column mappings, and hence enhance column-matching accuracy. An example of such techniques was proposed by Jaiswal et al. [46]. They developed a schema-matching tool that leverages value-mapping to enhance schema matching for schemas with opaque column names and opaque data instances for numerical and categorical columns. Opaque means “when it is difficult to understand the semantics of the data values [of a column] from its name” [46].

Exploiting Other Spreadsheet Features

Because scientists may use spreadsheets’ cell formulas and comments, we plan to incorporate this information into the process of relationship prediction. For instance, ReConnect might check if an added column contains derived data by first

checking if the values of that column are computed from formulas, and then determine whether these values are uniformly computed from other columns. Such an enhancement can also address the issue of formulas interfering with detecting relationships. ReDiscover could also use formulas to determine columns with comparable types. For instance, the formula $B4 + C4$ may indicate that columns B and C have comparable types. Additionally, ReDiscover could check if datasets match when computed columns are excluded, and if so it could suggest “equal on non-computed columns” relationship.

We can also exploit spreadsheet formulas in detecting related datasets. Even though in scientific spreadsheets formulas rarely contain references to cells in other sheets within a spreadsheet or in other spreadsheet documents, knowing such information can help ReDiscover identify datasets that may be related to the dataset being examined. For example, suppose we have three spreadsheets, $A.xlsx$, $B.xlsx$, and $C.xlsx$, each of which contains water temperature readings for a set of water samples at different points in time. Suppose we also have spreadsheet D , which uses the following formula to compute the average water temperature for this set of water samples stored in spreadsheets A , B , and C :

$$AVERAGE([A.xlsx]WSample1!B2, [B.xlsx]WSample2!B2, [C.xlsx]WSample3!B2)$$

From this formula, ReDiscover can determine the spreadsheet names ($A.xlsx$, $B.xlsx$, and $C.xlsx$), sheet names (WSample1, WSample2 and WSample3) and column addresses ($!$B2) of the related datasets. Such information can be used as features for the column-matching and the relationship-prediction classifiers.

7.1.4 Integrating ReConnet with ReDiscover

A future extension of our work is to combine ReDiscover with ReConnect to form a full relationship-identification system. Such a system would enable scientists

to predict and test for relationships in large collections of spreadsheets in an integrated environment. It would help scientists decide how to work with their datasets by suggesting possible operations on datasets based on discovered relationships. For instance, when the system identifies the *complementation* relationship between two datasets, it could suggest combining their data into one dataset via a join.

Furthermore, during the second part of our user study, several scientists suggested that ReConnect should combine tabular datasets or remove irrelevant datasets based on the suggested relationships. Thus, we plan to extend our methodology so that it suggests and follows a course of action based on the relationships that it identifies. For instance, the relationship-identification methodology might be able to join complementary information in a table, and create data views that reduce data duplication and complexity.

7.2 FURTHER APPLICATIONS

In this work we focus on detecting relationships that would result from the kinds of activities that scientists perform when operating on scientific datasets stored in spreadsheets. However, spreadsheets are also used to store other types of tabular datasets, such as statistics, budgets, and sales reports. It seems obvious to ask here whether our methods would work with such datasets. While ReDiscover can detect instances of the current relationships between data in such spreadsheets, the types of relationships that exist and their likelihood in other domains might differ from what is common in scientific spreadsheets.

We can separate the applicability of our techniques to spreadsheets from other domains into four categories: applications with similar relationships, applications with new relationships, applications with new relationships requiring new features, and applications that use different data organizations.

7.2.1 Applications with Similar Relationships

We could apply our system as is to applications where users perform activities on their datasets similar to those performed by scientists. User activities, such as adding or deleting rows or columns, filling in missing values, and reordering data would produce the same relationships, such as containment, augmentation, and duplicate, that exist between scientific datasets.

One such application could be in marketing where a group of market analysts is collecting a list of hotels to help a client make informed decision about establishing a new hotel. These analysts use spreadsheets to create, manipulate their list of hotels by adding or deleting rows (e.g., hotels) or columns (e.g., hotel rating, room types, price per night, etc.). They could use ReDiscover as is to detect relationships, such as row-containment, augmentation, and complementation, between their datasets. For example, they could use ReDiscover to detect the complementation relationship between their datasets to identify datasets that have complementary information about similar hotels.

7.2.2 Applications with New Relationships

Another class of application domains is where users perform actions that produce new kinds of relationships, but where we can still use the same column and spreadsheet-metadata features to detect these relationships. For such applications, we only need to update the Predict Relationship process of ReDiscover. We can do that simply by training new SVM models to predict these new relationships using new combinations of existing features.

An example of this domain is applications where users need to manage and analyze government data and statistics stored in spreadsheets. As a result of users' activities on these datasets, several new relationships could arise among them. For instance, a dataset about unemployment rates by state might extended by

adding new columns for new months. But there might be revised values for previous months, so the containment relationship does not hold between the original dataset and its extended versions. An economic analyst might want to determine if there are extended versions of the dataset he or she is currently working with. To detect such a relationship, we would train an SVM using combined features from augmentation and the various containment relationships to predict the new “revised” relationship.

Maintenance_V1		Maintenance_V2	
Item	Cost	Item	Cost
Roof Material	1500.40	Roof Material	1500
Window Frames	350.95	Window Frames	351
Doors	120.57	Doors	121
Woods	140.99	Woods	141
Paint	220.36	Paint	220
Labor	400.96	Labor	401

Figure 7.3: An example of two versions of a dataset about home maintenance costs, with a *near-match* relationship. The values of *Cost* column of dataset Maintenance_V1 has been rounded to the nearest dollar in Maintenance_V2 dataset.

In Figure 7.3, we show another example of two datasets with a *near-match* relationship, which is a relationship that was suggested by one of our user study participants (see Section 3.2.1). The rows of the *Cost* column of both Maintenance_V1 and V2 datasets are equal within \$1, because *Cost* column values from Maintenance_V1 has been rounded to the nearest dollar in Maintenance_V2. ReDiscover could use existing column summaries, such as mean and standard deviation, for detecting the near-match relationship.

7.2.3 Applications with New Relationships Requiring New Features

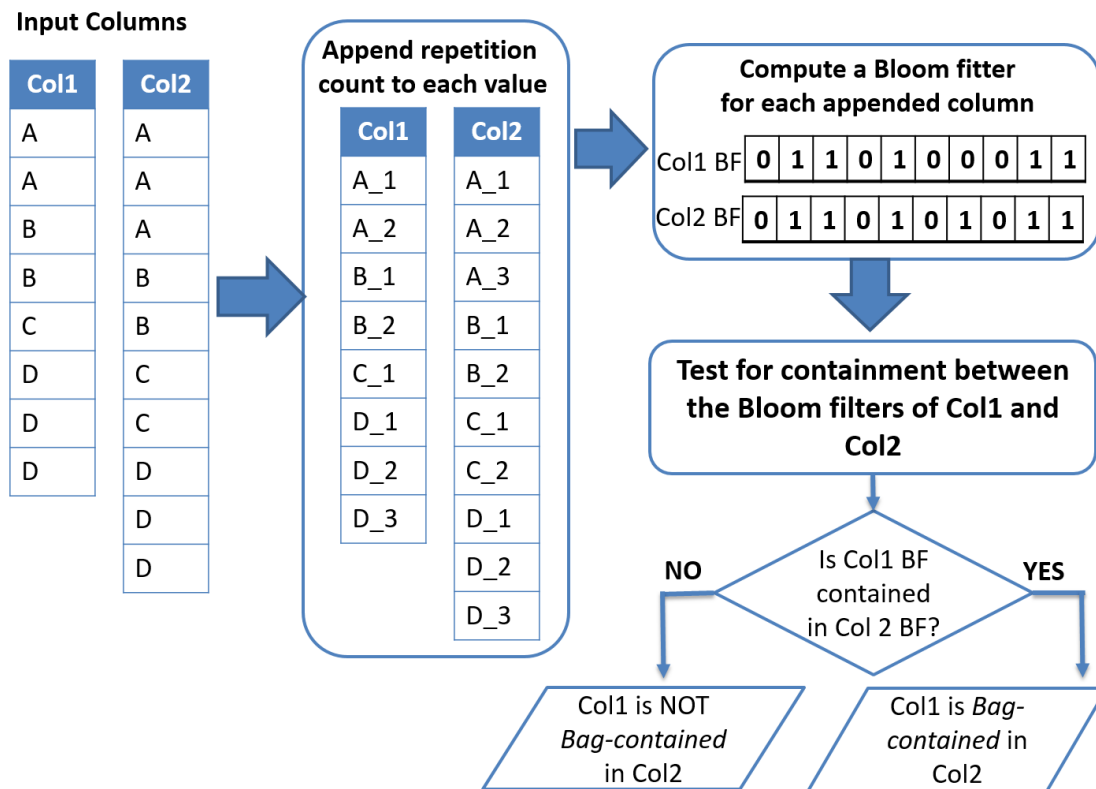


Figure 7.4: An example of using Bloom filters for computing indicative features for the *bag-row-containment* relationship.

There are applications where user actions could produce relationships that require new features. These features would require computing new column summaries, or adjusting column-matching or some other component of ReDiscover. An example of such domain is when users need to determine if a dataset with duplicate rows is row-contained in another dataset. This relationship is different from the set-based row-containment that ReDiscover predicts, as it is based on bag semantics (i.e., duplicate rows are considered). Thus, using the existing row-containment features is not enough for predicting the bag-based row-containment (*bag-row-containment*) relationship, as these features are based on Bloom filters,

which do not consider duplicates. Notice that if the *bag-row-containment* relationship holds between two datasets, then row-containment must hold as well. Thus, we would only need to predict the *bag-row-containment* between datasets with the row-containment relationship.

In Figure 7.4, we show how we can use Bloom filters for computing an indicative binary feature, *isBagContained*, for the *bag-row-containment* relationship. First, for each column, we append each of its values with the repetition count of that value in the column. Next, we compute a Bloom filter for each of the appended columns. Then, we test to see if the Bloom filters of dataset *A* columns are contained in their corresponding columns in *B*. Lastly, the *isBagContained* feature is set to one when all of *A*'s Bloom filters are contained in their corresponding Bloom filters of *B*'s columns.

7.2.4 Applications with Different Data Formats

There are also applications where the nature of the data is fundamentally different, and for which perhaps all of ReDiscover's components would have to be revised. Many such applications use spreadsheets that are developed when formatting data for human consumption. Often, these spreadsheets contain tabular datasets with row and column labels, and have subtotals, averages, and percentages at various cells in a column.

Figure 7.5 shows an example of a project-budget-report spreadsheet⁵ with row (Project Design, Project Development, etc.) and column (PROJECT TASKS, LABOR HOURS, etc.) labels. It also has row totals (TOTAL PER TASK), and column subtotals at various points in the columns. We might want to just extract the base data in each column and leaving behind the derived data.

The current version of ReDiscover extracts cell formulas, which could help

⁵Source: Microsoft Excel 2013 Templates. The "Project Budget" template.

	A	B	C	D	E	F	G	H	I
4			PROJECT TASKS	LABOR HOURS	LABOR COST (\$)	MATERIAL COST (\$)	TRAVEL COST (\$)	OTHER COST (\$)	TOTAL PER TASK
5		PROJECT DESIGN	Develop Functional Specifications	1.0	\$1.00	\$1.00	\$1.00	\$1.00	\$5.00
6	Develop System Architecture		1.0	\$1.00	\$1.00	\$1.00	\$1.00	\$5.00	
7	Develop Preliminary Design Specification		1.0	\$1.00	\$1.00	\$1.00	\$1.00	\$5.00	
8	Develop Detailed Design Specifications		1.0	\$1.00	\$1.00	\$1.00	\$1.00	\$5.00	
9	Develop Acceptance Test Plan		1.0	\$1.00	\$1.00	\$1.00	\$1.00	\$5.00	
10			Subtotal	5.0	\$5.00	\$5.00	\$5.00	\$5.00	\$25.00
11		PROJECT DEVELOPMENT	Develop Components	0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
12	Procure Software		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
13	Procure Hardware		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
14	Development Acceptance Test Package		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
15	Perform Unit/Integration Test		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
16			Subtotal	0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
17		PROJECT DELIVERY	Install System	0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
18	Train Customers		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
19	Perform Acceptance Test		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
20	Perform Post Project Review		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
21	Provide Warranty Support		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
22			Archive Materials	0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
23			Subtotal	0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
24		PROJECT MANAGEMENT	Customer Progress Meetings/Reports	0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
25	Internal Status Meetings/Reports		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
26	Third-Party Vendor Interface		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
27	Interface to Other Internal Departments		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
28	Configuration Management		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
29	Quality Assurance		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
30	Overall Project Management		0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	
31			Subtotal	0.0	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00

Figure 7.5: A sample spreadsheet with vertical and horizontal labels and subtotals.

distinguishing based from derived data. However, we still need to develop new routines for analyzing extracted formulas to distinguish between the two. We also need to update the Label Cells process of ReDiscover to handle row labels when labeling a dataset's cells.

Outside of the realm of spreadsheets, we want to explore the applicability of the relationship-identification methodology to tabular datasets originating in other settings, such as DBMSs. A straightforward way for our tool to detect the same relationships in ordered-table datasets that originate in other formats is to convert these formats into spreadsheet format. However, the relationships that may exist between datasets in different formats could be different in kind and likelihood based on the tools used to generate that format. For example, with spreadsheet tools, cutting and pasting a column is easy, but it is not as easy in a DBMS environment. In an DBMS, it is easy to remove rows according to a complex predicate, though that is not as easy with a spreadsheet tool. Furthermore, researchers tend to create

a spreadsheet for each run of an experiment or an observation session, whereas in a DBMS they would not necessarily create a new table. Consequently, certain relationships are likely to exist between datasets that were generated by one tool may not exist between those that were produced by another

7.3 CONCLUSION

File-based scientific datasets proliferate as a result of scientists' activities, such as sharing datasets with collaborators and receiving versions with modifications, and copying datasets for backup and analysis purposes. Consequently, scientists can lose track over time of how their datasets are connected. Hence it is challenging for them to determine which datasets are best for a given task or how to work with the data stored in their datasets. Frustrated by the complexity and the time it takes to manually identify connections among their datasets, especially in large collections, scientists may select outdated or incomplete datasets, or even forego the planned task. Existing scientific data-management-systems focus on managing datasets stored in DBMSs. However, the problem of managing file-based scientific datasets has not received much attention by the data-management research community.

In this dissertation, we present a relationship-identification methodology as a solution to this problem. We articulated a set of relationships, such as *augmentation*, *complementation*, and *duplicate*, that can help scientists determine the original connections among their datasets. To examine the feasibility of our approach, we developed ReConnect, a semi-automated tool for identifying relationships between two datasets. ReConnect relies on users to specify the location of a dataset in a spreadsheet, and to help with matching the columns of a dataset pair. Then, it suggests possible relationships based on how the columns of two datasets correspond. Users can select the relationship they would like to validate, and then ReConnect generates SQL queries to test for the relationship. Our user study results showed that subjects found ReConnect useful, and that determining relationships

with ReConnect was easier and less error-prone and time-consuming than visually inspecting their datasets. We also evaluated the effectiveness of ReConnect and four change-inference tools in identifying relationships between spreadsheets. We found change-inference tools are difficult to use for this task. ReConnect offers a “set at a time” approach for determining connections between spreadsheet pairs, while the other tools are row-, column- or cell-oriented — which does not scale as spreadsheets become larger. ReConnect works on a more abstracted level than the other tools, which can help users understand connections more easily.

Encouraged by the results of ReConnect’s evaluation, we extended our approach to handle collections of datasets, and developed a prototype system, ReDiscover, that automatically predicts, from a collection of datasets, the pairs that are likely related and the relationship between them. ReDiscover extracts data columns from spreadsheets, summarizes data in these columns by computing statistics and bit-vectors (Bloom filters), and uses these summaries to match columns of dataset pairs. Then, it uses column summaries, the column-correspondence and spreadsheet metadata to predict relationships between dataset pairs. Our evaluation of ReDiscover showed that it predicted relationships with good accuracy, and that it outperformed an approximated human-based approach, which encodes the strategies that a human might use to identify the tested relationships. We believe that we can further improve ReDiscover’s performance by implementing the enhancements we discussed in Section 7.1.

While this work focuses on the feasibility of automating our relationship-prediction methodology, and targets spreadsheet collections of individual scientists and small groups of researchers, our techniques are designed with scalability in mind. Scaling our relationship-prediction methodology to handle very large collections of datasets will broaden its applicability in other domains. Several colleagues in the data management community, who reviewed our work, saw the potential for applying our techniques to spreadsheets in other domains, such as business. One step in this

direction is to implement and experiment with the blocking techniques that we discussed in Section 7.1.2.

Nevertheless, our methodology already has significance, especially for scientists, who lack the technical skills or financial resources to use relational database systems. Such scientists rely heavily on spreadsheets, and need tools that can help them overcome impediments to sharing their data or deciding how to work with it. As one of the anonymous reviewers for the International Conference on Data Engineering stated “It is true that in enterprise and scientific settings, a large percentage of the actual data reside in spreadsheet[s]. So any solution that is able to extract these data and make them useful beyond the confines of the spreadsheet itself makes an important contribution.”

REFERENCES

- [1] Inter-university Consortium for Political and Social Research. <http://www.icpsr.umich.edu> (Visited on 2012/09/03).
- [2] The Research Data Alliance. <https://rd-alliance.org/> (Visited on 2013/01/14).
- [3] ABRAHAM, R., AND ERWIG, M. UCheck: A Spreadsheet Type Checker for End Users. *Visual Languages and Human-Centric Computing (VL/HCC), IEEE Symposium on 18*, 1 (Feb. 2007), 71–95.
- [4] ALAGIANNIS, I., BOROVICA-GAJIC, R., BRANCO, M., IDREOS, S., AND AILAMAKI, A. NoDB: Efficient Query Execution on Raw Data Files. *Communications of the ACM* 58, 12 (Nov 2015), 112–121.
- [5] ALAWINI, A., MAIER, D., TUFTE, K., AND HOWE, B. Helping Scientists Reconnect their Datasets. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management (2014)*, ACM, p. 29.
- [6] ALAWINI, A., MAIER, D., TUFTE, K., HOWE, B., AND NANDIKUR, R. Towards Automated Prediction of Relationships Among Scientific Datasets. In *Proceedings of the 27th International Conference on Scientific and Statistical Database Management (2015)*, ACM, pp. 35:1–35:5.
- [7] ALTINTAS, I., BARNEY, O., AND JAEGER-FRANK, E. Provenance Collection Support in the Kepler Scientific Workflow System. In *Provenance and Annotation of Data*, L. Moreau and I. Foster, Eds., vol. 4145 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 118–132.

- [8] BAROWY, D. W., GULWANI, S., HART, T., AND ZORN, B. FlashRelate: Extracting Relational Data from Semi-Structured Spreadsheets Using Examples. *Microsoft Research Technical Report*, MSR-TR-2014-53 (2014).
- [9] BELLAHSÉNE, Z., BONIFATI, A., AND RAHM, E. *Schema Matching and Mapping*. Springer, 2011.
- [10] BERNSTEIN, P., AND HAAS, L. Information Integration in the Enterprise. *Communications of the ACM* 51, 9 (2008), 72–79.
- [11] BLOOM, B. H. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications ACM Journal* 13, 7 (July 1970), 422–426.
- [12] BYUN, H., AND LEE, S. Applications of Support Vector Machines for Pattern Recognition: A Survey. In *Pattern Recognition with Support Vector Machines*. Springer, 2002, pp. 213–236.
- [13] CALLAHAN, S. P., FREIRE, J., SANTOS, E., SCHEIDEGGER, C. E., SILVA, C. T., AND VO, H. T. VisTrails: Visualization Meets Data Management. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data* (2006), ACM, pp. 745–747.
- [14] CENTER FOR DISEASE CONTROL AND PREVENTION. Resident Population, by Age, Sex, Race, and Hispanic Origin: United States, Selected Years 1950-2009, 2010.
- [15] CHAMBERS, C., AND ERWIG, M. Automatic Detection of Dimension Errors in Spreadsheets. *Visual Languages and Human-Centric Computing (VL/HCC), IEEE Symposium on* 20, 4 (2009), 269–283.
- [16] CHAMBERS, C., ERWIG, M., AND LUCKEY, M. SheetDiff: A Tool for Identifying Changes in Spreadsheets. In *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on* (2010), pp. 85–92.

- [17] CHANG, C., AND LIN, C. LIBSVM: a Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, 3 (2011), 27.
- [18] CHEN, Z., AND CAFARELLA, M. Automatic Web Spreadsheet Data Extraction. In *Proceedings of the 3rd International Workshop on Semantic Search Over the Web* (2013), ACM, p. 1.
- [19] CHEN, Z., CAFARELLA, M., CHEN, J., PREVO, D., AND ZHUANG, J. Senbazuru: A Prototype Spreadsheet Database Management System. *Proceedings of the VLDB Endowment* 6, 12 (2013), 1202–1205.
- [20] CORTES, C., AND VAPNIK, V. Support-vector Networks. *Machine learning* 20, 3 (1995), 273–297.
- [21] CUDRE-MAUROUX, P., KIMURA, H., LIM, K.-T., ROGERS, J., SIMAKOV, R., SOROUSH, E., VELIKHOV, P., WANG, D. L., BALAZINSKA, M., BECLA, J., DEWITT, D., HEATH, B., MAIER, D., MADDEN, S., PATEL, J., STONEBRAKER, M., AND ZDONIK, S. A Demonstration of SciDB: A Science-oriented DBMS. *The Proceedings of the VLDB Endowment (PVLDB)* 2, 2 (Aug. 2009), 1534–1537.
- [22] CUNHA, J., ERWIG, M., AND SARAIVA, J. Automatically Inferring ClassSheet Models from Spreadsheets. In *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on* (2010), pp. 93–100.
- [23] CUNHA, J., SARAIVA, J., AND VISSER, J. Discovery-based Edit Assistance for Spreadsheets. In *Visual Languages and Human-Centric Computing, 2009. VL/HCC 2009. IEEE Symposium on* (Sept 2009), pp. 233–237.

- [24] CUNHA, J., SARAIVA, J., AND VISSER, J. From Spreadsheets to Relational Databases and Back. In *Proceedings of the 2009 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (2009)*, ACM, pp. 179–188.
- [25] DASU, T., JOHNSON, T., MUTHUKRISHNAN, S., AND SHKAPENYUK, V. Mining Database Structure; or, How to Build a Data Quality Browser. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (2002)*, SIGMOD '02, ACM, pp. 240–251.
- [26] DAVIDSON, S. B., BOULAKIA, S. C., EYAL, A., LUDÄSCHER, B., MCPHILLIPS, T. M., BOWERS, S., ANAND, M. K., AND FREIRE, J. Provenance in Scientific Workflow Systems. *IEEE Data Engineering 30*, 4 (2007), 44–50.
- [27] DE OLIVEIRA, D., OGASAWARA, E., BAIAO, F., AND MATTOSO, M. SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on (July 2010)*, pp. 378–385.
- [28] DICE, L. R. Measures of the Amount of Ecologic Association between Species. *Ecology 26*, 3 (1945), 297–302.
- [29] DOAN, A., DOMINGOS, P., AND HALEVY, A. Reconciling Schemas of Disparate Data Sources: A Machine-learning Approach. *ACM SIGMOD Record* (2001).
- [30] DOAN, A., DOMINGOS, P., AND LEVY, A. Learning Source Descriptions for Data Integration. In *Informal Proceedings of the International Workshop on the Web and Databases, WebDB (2000)*, pp. 81–86.

- [31] DONG, B., BYNA, S., AND WU, K. SDS: A Framework for Scientific Data Services. In *Proceedings of the 8th Parallel Data Storage Workshop (2013)*, PDSW '13, ACM, pp. 27–32.
- [32] ELMAGARMID, A., IPEIROTIS, P., AND VERYKIOS, V. Duplicate Record Detection: A Survey. *Knowledge and Data Engineering, IEEE Transactions on* 19, 1 (2007), 1–16.
- [33] FISHER, M., AND ROTHERMEL, G. the EUSES Spreadsheet Corpus: a Shared Resource for Supporting Experimentation with Spreadsheet Dependability Mechanisms. In *ACM SIGSOFT Software Engineering Notes (2005)*, vol. 30, ACM, pp. 1–5.
- [34] FLORENCESOFT. DiffEngineX: Compare Excel Work Sheets, 2010. <http://www.florencesoft.com/compare-excel-workbooks-differences.html> (Visited on 2013/07/25).
- [35] FOSTER, I., VOCKLER, J., WILDE, M., AND ZHAO, Y. Chimera: a Virtual Data System for Representing, Querying, and Automating Data Derivation. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management.* (2002), pp. 37–46.
- [36] GETOOR, L., AND MACHANAVAJJHALA, A. Entity Resolution for Big Data. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2013)*, KDD '13, ACM, pp. 1527–1527.
- [37] GITHUB. GitHub: Smarter Version Control. <https://github.com/> (Visited on 2014/03/14).
- [38] GONZALEZ, H., HALEVY, A., JENSEN, C., LANGEN, A., MADHAVAN, J., SHAPLEY, R., SHEN, W., AND GOLDBERG-KIDON, J. Google Fusion Tables: Web-centered Data Management and Collaboration. In *Proceedings of*

the 2010 ACM SIGMOD International Conference on Management of Data (2010), pp. 1061–1066.

- [39] GRAVANO, L., IPEIROTIS, P., KOUDAS, N., AND SRIVASTAVA, D. Text Joins in an RDBMS for Web Data Integration. In *Proceedings of the 12th International Conference on the World Wide Web* (2003), ACM, pp. 90–101.
- [40] HDF GROUP AND OTHERS. Hierarchical Data Format, version 5, 2014. <https://www.hdfgroup.org/> (Visited on 2016/01/06).
- [41] HERNÁNDEZ, M., MILLER, R., AND HAAS, L. Clio: A Semi-automatic Tool for Schema Mapping. *SIGMOD Rec.* 30, 2 (2001), 607.
- [42] HOWE, B., COLE, G., SOURUSH, E., KOUTRIS, P., KEY, A., KHOUSAINOVA, N., AND BATTLE, L. Database-as-a-Service for Long-Tail Science. In *Scientific and Statistical Database Management*. 2011, pp. 480–489.
- [43] ISARD, M., BUDIU, M., YU, Y., BIRRELL, A., AND FETTERLY, D. Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. *ACM SIGOPS Operating Systems Review* 41, 3 (2007), 59–72.
- [44] IVANOVA, M., KERSTEN, M., MANEGOLD, S., AND KARGIN, Y. Data Vaults: Database Technology for Scientific File Repositories. *Computing in Science and Engineering* 15, 3 (2013), 32–42.
- [45] JAIN, N., DAHLIN, M., AND TEWARI, R. Using Bloom Filters to Refine Web Search Results. In *The 8th International Workshop on the Web and Databases (WebDB)* (2005), pp. 25–30.
- [46] JAISWAL, A., MILLER, D., AND MITRA, P. Schema Matching and Embedded Value Mapping for Databases with Opaque Column Names and Mixed Continuous and Discrete-valued Data Fields. *ACM Transactions on Database Systems (TODS)* 38, 1 (2013), 2.

- [47] KIMBALL, R., ROSS, M., THORNTHWAITTE, W., MUNDY, J., AND BECKER, B. *The Data Warehouse Lifecycle Toolkit*. Wiley, 2011.
- [48] KUDO, T. CRF++: Yet Another CRF Toolkit. *Software available at <http://crfpp.sourceforge.net>* (2005). (Visited on 2013/01/15).
- [49] LAFFERTY, J., MCCALLUM, A., AND PEREIRA, F. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning* (2001), ICML '01, Morgan Kaufmann Publishers Inc., pp. 282–289.
- [50] LEVENSHTAIN, V. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In *Soviet Physics Doklady* (1966), vol. 10, p. 707.
- [51] LI, W., AND CLIFTON, C. SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Networks. *Data & Knowledge Engineering* 33, 1 (April 2000), 49–84.
- [52] LI, W., CLIFTON, C., AND LIU, S. Database Integration Using Neural Networks: Implementation and Experiences. *Knowledge and Information Systems* 2, 1 (2000), 73–96.
- [53] LI, W.-S., AND CLIFTON, C. Semantic Integration in Heterogeneous Databases Using Neural Networks. In *Proceedings of the 20th International Conference on Very Large Data Bases* (1994), pp. 1–12.
- [54] MACEFIELD, R. How to Specify the Participant Group Size for Usability Studies: A Practitioner’s Guide. *Journal of Usability Studies* 5, 1 (2009), 34–45.
- [55] MICHENER, W., VIEGLAIS, D., VISION, T., KUNZE, J., CRUSE, P., AND JANÉE, G. DataONE: Data Observation Network for Earth-preserving Data

- and Enabling Innovation in the Biological and Environmental Sciences. *D-Lib Magazine* 17, 1 (2011), 3.
- [56] MICROSOFT. What You Can Do with Spreadsheet Inquire, 2013. <http://office.microsoft.com/en-us/excel-help/what-you-can-do-with-spreadsheet-inquire-HA102835926.aspx> (Visited on 2013/11/20).
- [57] OUNIS, I., AMATI, G., PLACHOURAS, V., HE, B., MACDONALD, C., AND LIOMA, C. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings of the OSIR Workshop* (2006), pp. 18–25.
- [58] PAPADAKIS, G., PAPASTEFANATOS, G., AND KOUTRIKA, G. Supervised Meta-blocking. *Proceedings of the VLDB Endowment* 7, 14 (2014), 1929–1940.
- [59] PINTO, D., MCCALLUM, A., WEI, X., AND CROFT, B. Table Extraction Using Conditional Random Fields. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2003), ACM, pp. 235–242.
- [60] RAHM, E., AND BERNSTEIN, P. A Survey of Approaches to Automatic Schema Matching. *the VLDB Journal* 10, 4 (2001), 334–350.
- [61] RAHM, E., AND DO, H. Data Cleaning: Problems and Current Approaches. *IEEE Data Eng. Bull.* 23, 4 (2000), 3–13.
- [62] RAM, K. Git can Facilitate Greater Reproducibility and Increased Transparency in Science. *Source Code for Biology and Medicine* 8, 1 (2013), 7.
- [63] REW, R., AND DAVIS, G. NetCDF: an Interface for Scientific Data Access. *IEEE Transactions on Computer Graphics and Applications* 10, 4 (1990), 76–82.

- [64] ROSTIN, A., ALBRECHT, O., BAUCKMANN, J., NAUMANN, F., AND LESER, U. A Machine Learning Approach to Foreign Key Discovery. In *The 12th International Workshop on the Web and Databases, WebDB* (2009).
- [65] SCHNELL, R., BACHTELER, T., AND REIHER, J. Privacy-Preserving Record Linkage using Bloom Filters. *BMC Medical Informatics and Decision Making* 9, 1 (2009), 1–11.
- [66] SCHOPF, J. M. Treating data like software: a case for production quality data. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries* (2012), ACM, pp. 153–156.
- [67] SLASHDOT. SorceForge: Find, Create, and Publish Open Source Software for Free. <http://sourceforge.net/> (Visited on 2014/03/14).
- [68] STAROBINSKI, D., TRACHTENBERG, A., AND AGARWAL, S. Efficient PDA Synchronization. *Mobile Computing, IEEE Transactions on* 2, 1 (2003), 40–51.
- [69] STEVENS, R. D., ROBINSON, A. J., AND GOBLE, C. A. MyGrid: Personalised Bioinformatics on the Information Grid. *Bioinformatics* 19, suppl 1 (2003), i302–i304.
- [70] SUTTON, C., AND MCCALLUM, A. *Introduction to Conditional Random Fields for Relational Learning*. MIT Press, 2006.
- [71] TREK. Common Evaluation Measures, 2006.
<http://trec.nist.gov/pubs/trec15/appendices/CE.MEASURES06.pdf> (Visited on 2014/03/20).
- [72] VENETIS, P., HALEVY, A., MADHAVAN, J., PAŞCA, M., SHEN, W., WU, F., MIAO, G., AND WU, C. Recovering Semantics of Tables on the Web. *Proc. VLDB Endow.* 4, 9 (2011), 528–538.

- [73] WELLS, D. C., GREISEN, E., AND HARTEN, R. FITS-a Flexible Image Transport System. *Astronomy and Astrophysics Supplement Series 44* (1981), 363.
- [74] WYATT, J., AND FERGUSON, E. Spreadsheets Risk Management: Frequently Asked Questions Guide. *Protiviti* (2011), Technical Report FAQ09. <http://www.protiviti.com/en-UK/Documents/UK-Spreadsheet-Risk-Management-FAQ.pdf>.
- [75] XL-CONSULTING. Synkronizer: Compares Excel Files Faster than You Can, 2010. <http://www.synkronizer.com/> (Visited on 2013/07/23).
- [76] ZARDETTO, D., SCANNAPIECO, M., AND CATARCI, T. Effective Automated Object Matching. In *Data Engineering, 2010 IEEE 26th International Conference on* (2010), IEEE, pp. 757–768.

APPENDIX A: CELL FEATURES

Cell Context Features

Feature ID	Description
<i>is_left_alpha</i>	Does the left cell contain alphabetical characters only?
<i>is_left_num</i>	Does the left cell contain numbers only?
<i>is_left_alphanum</i>	Does the left cell contain alphanumerics?
<i>is_left_header</i>	Is the left cell content in the list of column headers?
<i>is_left_empty</i>	Is the left cell empty?
<i>is_right_alpha</i>	Does the right cell contain alphabetical characters only?
<i>is_right_num</i>	Does the right cell contain numbers only?
<i>is_right_alphanum</i>	Does the right cell contain alphanumerics?
<i>is_right_header</i>	Is the right cell content in the list of column headers?
<i>is_right_empty</i>	Is the right cell empty?
<i>is_above_alpha</i>	Does the cell above contain alphabetical characters only?
<i>is_above_num</i>	Does the cell above contain numbers only?
<i>is_above_alphanum</i>	Does the cell above contain alphanumerics?
<i>is_above_header</i>	Is the cell above content in the list of column headers?

<i>is_left_empty</i>	Is the cell above empty?
<i>is_below_alpha</i>	Does the cell below contain alphabetical characters only?
<i>is_below_num</i>	Does the cell below contain numbers only?.
<i>is_below_alphanum</i>	Does the cell below contain alphanumerics?
<i>is_below_header</i>	Is the cell below content in the list of column headers?
<i>is_below_empty</i>	Is the cell below empty?

Cell Layout Features

Feature ID	Description
<i>is_merge_cell</i>	Is the cell merged with other cells?
<i>is_left_align.</i>	Is the cell aligned to the left?
<i>is_right_align.</i>	Is the cell aligned to the right?
<i>is_center_align.</i>	Is the cell aligned to the center?
<i>is_font_italic</i>	Is the cell font italicized?
<i>is_font_underlined</i>	Is the cell font underlined?
<i>is_font_bold</i>	Is the cell font bold?

Cell Text Features

Feature ID	Description
<i>is_alpha</i>	Does the cell contain only alphabetical characters?
<i>is_alpha</i>	Does the cell contain only numerical characters?
<i>is_alphanum</i>	Does the cell contain alphanumerical characters?
<i>is_empty</i>	Is the cell empty?
<i>is_all_small</i>	Are all alphabetical characters small?
<i>is_all_Capital</i>	Are all alphabetical characters capital?
<i>starts_capital</i>	Is the first character capital?

Cell Content Features

Feature ID	Description
<i>is_in_nulls</i>	Is the cell content in the list of default nulls?
<i>is_in_headers</i>	Is the cell content in the list of column header terms?
<i>contain_colon</i>	Does the cell contain a colon?
<i>contain_special</i>	Does the cell contain any special characters?
<i>is_long_text</i>	Does the cell contain more than 40 characters?
<i>in_year_range</i>	Is the cell content a number between 1900 and 2050?

APPENDIX B: REDISCOVER'S COLUMN SUMMARIES

Description of the column summaries collected by ReDiscover.

Summary Name	Description
<i>column_id</i>	The column header name and column location information: spreadsheet and sheet names, the order of the column, and the address of the first and last cell in that column
<i>col_type</i>	The inferred data type of the column
<i>row_count</i>	The count of row values of the specified column
<i>unique</i>	The count of unique values of the specified column
<i>null</i>	The count of null values of the specified column
<i>max_val</i>	The maximum value of the specified column
<i>min_val</i>	The minimum value of the specified column
<i>mean</i>	The average of the values (for numerical columns) of the specified column
<i>Std_dev</i>	The standard deviation of the values (for numerical columns) of the specified column <i>A</i> and <i>B</i> .
<i>common_val_0</i>	The first most common value in the specified column
<i>cv0_freq</i>	The frequency count of the first most common value in the specified column
<i>common_val_1</i>	The second most common value in the specified column

<i>cv1_freq</i>	The frequency count of the second most common value in the specified column
<i>common_val_2</i>	The third most common value in the specified column
<i>cv2_freq</i>	The frequency count of the third most common value in the specified column
<i>common_val_3</i>	The fourth most common value in the specified column
<i>cv3_freq</i>	The frequency count of the fourth most common value in the specified column
<i>common_val_4</i>	The fifth most common value in the specified column
<i>cv4_freq</i>	The frequency count of the fifth most common value in the specified column
<i>common_val_5</i>	The sixth most common value in the specified column
<i>cv5_freq</i>	The frequency count of the sixth most common value in the specified column
<i>common_val_6</i>	The seventh most common value in the specified column
<i>cv6_freq</i>	The frequency count of the seventh most common value in the specified column
<i>common_val_7</i>	The eighth most common value in the specified column
<i>cv7_freq</i>	The frequency count of the eighth most common value in the specified column

<i>common_val_8</i>	The ninth most common value in the specified column
<i>cv8_freq</i>	The frequency count of the ninth most common value in the specified column
<i>common_val_9</i>	The tenth most common value in the specified column
<i>cv9_freq</i>	The frequency count of the tenth most common value in the specified column

APPENDIX C: HUMAN-BASED APPROACH FEATURES

Description of the human-based approach features.

Feature Name	Description
<i>columnNamesMatch</i>	Do the column names of dataset <i>A</i> match those of <i>B</i> ? (0: no matching column names; 1: some column names match; 2: all column names match)
<i>columnOrderMatch</i>	When the column names of datasets <i>A</i> and <i>B</i> match, are their columns have in same order?
<i>isWorkbookMatch</i>	Do datasets <i>A</i> and <i>B</i> have the same workbook name?
<i>isWorksheetMatch</i>	Do datasets <i>A</i> and <i>B</i> have the same worksheet name?
<i>isAuthorMatch</i>	Do datasets <i>A</i> and <i>B</i> have the same author name?
<i>isSizeEqual</i>	Is the size of datasets <i>A</i> and <i>B</i> equal?
<i>isRowCountEqual</i>	Is the row count of datasets <i>A</i> and <i>B</i> equal?
<i>isColCountEqual</i>	Is the column count of datasets <i>A</i> and <i>B</i> equal?
<i>cntMatchingMetadata</i>	The count of matching metadata between datasets <i>A</i> and <i>B</i> .
<i>isRowCountALessThanB</i>	Is the row count of dataset <i>A</i> less than that of <i>B</i> ?
<i>isRowCountBLessThanA</i>	Is the row count of dataset <i>B</i> less than that of <i>A</i> ?
<i>dataTypesMatch</i>	Do the columns of datasets <i>A</i> and <i>B</i> have the same data types?