**CAS 764 *&* CAS 771**

**2020 WINTER TERM**

**FINAL PROJECT REPORT**

**COMPARISON OF ENTITY MATCHING SYSTEMS**

**Baran Kaya**

400284996

kayab@mcmaster.ca

**CONTENT**

[ABSTRACT 3](#_Toc37768792)

[1. INTRODUCTION 3](#_Toc37768793)

[2. RELATED WORK 4](#_Toc37768794)

[3. DATASETS 5](#_Toc37768795)

[3.1. Bikes Dataset 5](#_Toc37768796)

[3.2. Restaurants Dataset 5](#_Toc37768797)

[3.3. Citations Dataset 6](#_Toc37768798)

[3.4. Products Dataset 7](#_Toc37768799)

[4. SYSTEMS/TOOLS 8](#_Toc37768800)

[4.1. Selected Systems 8](#_Toc37768801)

[4.2. Systems & Features 8](#_Toc37768802)

[4.2.1. Magellan 9](#_Toc37768803)

[4.2.2. SERF 10](#_Toc37768804)

[4.2.3. FRIL 11](#_Toc37768805)

[4.2.4. Febrl 14](#_Toc37768806)

[4.2.5. Data Ladder 16](#_Toc37768807)

[4.2.6. WinPure Clean & Match 19](#_Toc37768808)

[4.2.7. Management Ware Data Cleansing & Matching 21](#_Toc37768809)

[4.3. Who uses EM systems 23](#_Toc37768810)

[5. EXPERIMENTS 23](#_Toc37768811)

[5.1. System Specs 24](#_Toc37768812)

[5.2. Experiment Results 24](#_Toc37768813)

[5.2.1. Runtimes 24](#_Toc37768814)

[5.2.2. Accuracy 25](#_Toc37768815)

[5.3. Evaluation 40](#_Toc37768816)

[5.3.1. Precision 41](#_Toc37768817)

[5.3.2. Recall 42](#_Toc37768818)

[5.3.3. F1 42](#_Toc37768819)

[6. CONCLUSION 43](#_Toc37768820)

[7. REFERENCES 44](#_Toc37768821)

# ABSTRACT

Entity matching is one of the top research topics in the data management area. There are lots of different algorithms that do the same job. However, most of these research algorithms don’t have any implementation to use. There are only a few implemented entity matching algorithms or systems. In this report, research-based and commercial entity matching systems compared based on result accuracy, runtime performance, system usability and system features.

# INTRODUCTION

Entity Matching (EM) defines detecting the same real-world entities in the database systems. It is one of the hot topics in the Data Management area because data is getting bigger and bigger every day. Storing the same entity’s data more than once takes unnecessary space in the database. This problem both increases the database management cost and decreases the overall query performance of the database system. Entity matching has two different types: Deduplication and Record Linkage. Deduplication is detecting the same entities inside one dataset however linkage or record linkage means finding these same entities across multiple datasets. Deduplication compares tuples in the same dataset but record linkage compares different dataset’s tuples to find the same entity.

There are numerous algorithms to solve entity matching problems in databases and each of them tries to outperform previous ones on performance or accuracy metrics. Since there are a lot of algorithms, some researchers try to find the best one regarding the algorithm performance. Companies with EM commercial systems want to sell their products, therefore; these companies try to create easy to use and better-performing systems. On the other hand, research-based algorithms or systems try to improve previous ones’ performance metrics on runtime or accuracy. Finding the best EM system requires testing every one of them in the same environment with the same datasets. There are few survey papers which compare entity matching algorithms. However, all of the papers only use research-based algorithms in comparison. In this project paper, research-based implemented systems compared with commercial systems to figure out which is better. This is an important subject because there aren’t any comparisons between research and commercial systems. At the end of the paper, you will find out what are their most important features.

In this paper, every system is compared in four aspects. These aspects are results accuracy, runtime performance, system usability and additional system features. Each of the metrics will be explained in the experimental section *(Section 5)*. Since accuracy and the runtime performance will be compared, different datasets used for better comparison between these two aspects. Four different datasets used in the experiments. Some of the datasets have over a million tuples and some of them have nearly twenty attributes for comparing the similarity between tuples. Each dataset would show different aspects of the systems’ performance. Every dataset’s number of tuples, its attributes and why were they chosen explained in the dataset section *(Section 3).*

Related work about entity matching comparison mentioned in Section 2. Section 3 explains each datasets and their features. Section 4 has system explanations and why they were chosen over other entity matching systems. Section 5 is for experiments and evaluation which consists of system specs that all of the experiments run, experimental graph results and finally evaluation of each experiment. In Section 6, all of the sections will be summarized.

# RELATED WORK

Entity matching is a very important topic in the data management field. That’s why there are lots of different papers on this subject. Most of the papers present a new algorithm that has better performance scores on some of the metrics. There are a few comparisons and survey papers about entity matching, deduplication and linkage. Entity matching papers mostly focus on the performance (accuracy or runtime) of the algorithms, however; some papers focus on the security aspect of it. For instance, Patil and Dhanushree (2015) focuses only on security and encryption parts of the deduplication methods in the cloud systems.

Since, there are lots of algorithms around, there are few survey papers that compare these algorithms in the same conditions. There are some survey paper examples in literature (Baraterio and Galhardas, 2005; Bharambe et al., 2012; Christen, 2011; Elmagarmid et al., 2006; Köpcke and Rahm, 2010). These five papers compare different aspects of the entity matching algorithms. For example, Christen (2011) compares different indexing techniques such as blocking, sorted neighbor and clustering. These methods used for reducing the tuple-to-tuple comparison numbers so that performance could increase. Christen (2011) uses reduction ratio, pair quality and pair completeness metrics with dirty and clean datasets to evaluate the experiment results.

Bharambe et al. (2012) and Elmagarmid et al. (2006) papers are very similar in terms of content. Both of them explain the deduplication concepts and methods. These papers mention string distance methods like Jaro-Wrinkler, Smith-Waterman, Q-gram, edit distance and token-based distance metrics like Q-gram, atomic strings, WHIRL and phonetic distance metrics such as Soundex, ONCA. They also refer several deduplication techniques like rule-based, active-learning, probabilistic, unsupervised learning, indexing and similarity-distance… Lastly, they briefly touch on a few deduplication systems like Febrl, TAILOR, WHIRL and some database servers.

One paper’s main aim isn’t just entity matching but data quality. Baraterio and Galhardas, (2005) survey paper compares different data quality systems. This paper briefly explains different data quality problems. It mentions problems like conflicts, empty values, incorrect types, misspellings, encoding errors… Then the authors mention metrics of the system comparison. Some of the metrics are input data types, extraction capabilities, profiling, data transforming, system updates, interface, versioning, function libraries, debugging, etc. This paper doesn’t test the performance of the systems rather it checks thirty-seven commercial and research-based systems to if they have the above-listed metrics.

The last and the most similar research paper is written by Köpcke and Rahm (2010). This paper compares research-based entity matching frameworks. Functional comparison metrics used in Köpcke and Rahm’s (2010) paper are entity type (input), blocking methods, different matchers, combination of the matchers and training selection. Eleven different frameworks compared with these features and result effectiveness. Some of the frameworks used in both Köpcke and Rahm (2010)’s paper and this paper are SERF and Febrl systems. The paper also contains a comparison between manual, machine learning and hybrid approaches in the matchers.

# DATASETS

This project’s aim is to compare different systems that detect entity matching. Therefore, using datasets that specifically designed for entity matching is very important for experiment results. For that purpose, Magellan system’s example datasets used. On Magellan website there are four different sourced entity matching ready datasets. The biggest one (The 748 Datasets) has twenty-four different datasets and each of them has different domains and row numbers. However, the 748 datasets don’t have any match result files. The Corleone and The Falcon datasets have match result files that can be used for calculating the systems match accuracy. For this project, four different datasets have been chosen from different domains. Next subsections explain each of the datasets and their attributes. Each dataset’s tables show attribute features like name and type in addition, weight and threshold values for distance/similarity methods.

***Note:*** *Magellan has 2 tables for every dataset because Magellan is a record linkage (EM) system.*

## Bikes Dataset

Bikes dataset consists of online sales bikes. It has two tables and each table represents different online bike sale websites. Both sale websites from India. This dataset is chosen because both tables have more than 4K tuples. Also, both tables have the same attributes and most of the attributes are used in duplication detection matchers. Table 1 shows the Bikes dataset attribute features.

Number of tuples in the tables:

* Bike Dekho: 4785
* Bike Wale: 9002

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | Id | Bike Name | City Posted | KM Driven | Color | Fuel Type | Price | Model Year | Owner Type | URL |
| **Type** | Int | Str | Str | Int | Str | Str | Int | Int | Str | Str |
| **Weight** | - | 20% | 12% | 12% | 12% | 12% | 15% | 12% | 5% | - |
| **Threshold** | - | 80% | 90% | 90% | 95% | 90% | 90% | 95% | 70% | - |

Table 1: Bikes Dataset Attributes

## Restaurants Dataset

Magellan website has five different restaurants datasets. This restaurant dataset is the only one that has the matched tuples data file. Moreover, this dataset has the smallest one between all of them. Table 2 shows the Restaurants dataset attribute features.

Number of tuples in tables:

* Fodors: 533
* Zagats: 331
* Matches: 112

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | Id | Name | City | Address | Phone | Type | Class |
| **Type** | Int | Str | Str | Str | Str | Str | Int |
| **Weight** | - | 20% | 20% | 20% | 20% | 20% | - |
| **Threshold** | - | 80% | 90% | 70% | 85% | 90% | - |

Table 2: Restaurants Dataset Attributes

Restaurants dataset’s City attribute values are mostly similar. However, some tuples use city abbreviations such as LA (Los Angeles) or NYC (New York City). In this case, 90% similarity threshold wouldn’t work. 90% threshold works for typos or other minimal errors.

## Citations Dataset

Like restaurant datasets, citations have different datasets on the Magellan website. The dataset selected for this project has the largest number of tuples. Table 3 shows the Citations dataset attribute features.

Number of tuples in tables:

* Citeseer: 1823978
* DBLP: 2512927
* Matches: 558787

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | Id | Title | Authors | Journal | Month | Year | Publication Type |
| **Type** | Int | Str | Str | Str | Int | Int | Str |
| **Weight** | - | 70% | 30% | - | - | - | - |
| **Threshold** | - | 85% | 70% | - | - | - | - |

Table 3: Citations Dataset Attributes

This dataset consists of many columns; however, some of the columns have more than one value. For instance, the authors attribute may have more than one-person name. There are lots of tuples like that and these names separated with a comma character. The problem is, these datasets stored in a CSV (Comma Separated Values) format and some columns use commas to separate values. In most cases each column values inside the quotation marks but some of the tuples don’t have quotation marks. Therefore, all of the citation datasets in the Magellan website has problems with column values. Because both columns and values inside columns separated with commas. You can see the normal and faulty tuple examples below. Also, Figure 1 and Figure 2 indicate the dirtiness of the Citations dataset. Some columns are mostly empty and some columns’ values are incorrect.

* 1,”Title 1”,”John Smith, Ed Grey”,”Journal 1”,4,2005,ONLINE 🡪 Clean Data
* 2,Title2,John Smith,Ed Grey,Journal,4,2005,ONLINE 🡪 Dirty Data

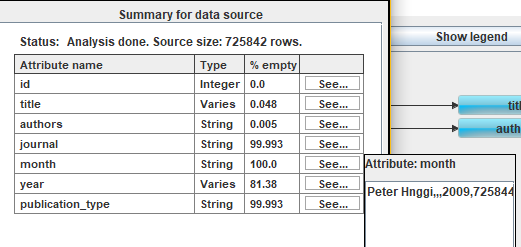


Figure 1: Citations Dataset Attribute Profile & Month Value in FRIL

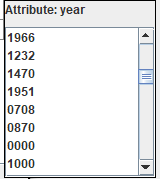
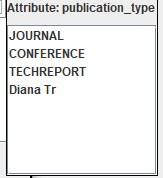
 

Figure 2: Citation Dataset Year & Publication Type Tuples in FRIL

## Products Dataset

This dataset consists of Amazon and Walmart online products. Amazon table is significantly bigger than the Walmart one and they have different column names and orders. They both have more than twenty columns/attributes. Only important and common attributes are shown in Table 4.

Number of tuples in tables:

* Amazon: 22074
* Walmart: 2554
* Matches: 1154

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | Id | Title | Price | Brand | Categories | Descript-ion | URL | Dimens-ions | Weight | Details | Model No |
| **Dataset** | Both | Both | Both | Both | Amazon | Walmart | Both | Both | Both | Amazon | Both |
| **Type** | Int | Str | Int | Str | Str | Str | Str | Str | Int | Str | Str |
| **Weight** | - | 25% | 35% | 25% | - | - | - | 2% | 3% | - | 10% |
| **Threshold** | - | 75% | 90% | 90% | - | - | - | 70% | 70% | - | 85% |

Table 4: Products Dataset Attributes

# SYSTEMS/TOOLS

## Selected Systems

The first step of this project was choosing the EM systems that will be tested. The importance of this project was comparing research-based systems with commercial systems. So, a total of 6-8 systems required. The first step was finding the best researched-based systems. For that reason, entity matching papers scanned to find the algorithms with implemented systems. Magellan [8] was one of the first EM systems that combine numerous matching features under a roof. That is why it was the first picked one. SERF [2] was chosen over some of the algorithms because it has implemented Java Framework that could be used in this project. Then, I found a list of open-sourced EM systems and checked some of them. In that list there were more than ten different EM implementations but, only a few of them had algorithm papers. FRIL [7] and Febrl [4] were chosen because they had entity matching algorithm papers. Later, the project needed commercial systems. There were more than five systems that can perform entity matching. However, some of the systems like IBM InfoSphere or SAS DataFlux don’t have free trial versions. That’s why Data Ladder and WinPure Clean & Match have chosen over some of the biggest EM systems. All of the systems that were used in this project listed in *Table 5*.

In summary, Magellan was chosen because it was the first research-based EM system. SERF was chosen due to its algorithm and implementation. FRIL and Febrl were chosen for their open-source background. Finally, Data Ladder, WinPure and Management Ware were chosen becuase their free trial/demo versions that were available to everyone.

|  |  |
| --- | --- |
| **Research Based Systems** | **Commercial Systems** |
| Magellan [8] | Data Ladder |
| SERF [2] | WinPure Clean & Match |
| FRIL [7] | Management Ware  Data Cleansing & Matching |
| Febrl [4] |  |

Table 5: Selected Systems

## Systems & Features

This section mostly focuses on the usability and features of the systems. The other two aspects, runtime performance and result accuracy, will be discussed in the *Section 5 Experiments & Evaluation*.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Features*  *Systems* | **GUI** | **# Input types** | **# Index methods** | **# Compare methods** | **Debugger** | **Profiling** | **Require-ments** |
| **Magellan** | No | 1 | 5 | 7 | Yes | Yes | Python |
| **SERF** | No | 1 | None | 3 | No | No | JDK |
| **FRIL** | Yes | 4 | 5 | 8 | Yes | Yes | JRE |
| **Febrl** | Yes | 3 | 7 | 26 | No | Yes | Python |
| **Data Ladder** | Yes | 5+ | None | 4 | No | Yes | - |
| **WinPure** | Yes | 4 | None | 4 | No | Yes | - |
| **Man. Ware** | Yes | 4 | None | None | No | Yes | - |

Table 6: All Systems Features

### Magellan

Magellan is a research-based entity matching (record linkage) system. It is one of the first systems that combine different abilities under a roof. It uses Python as a primary language and its basically a Python library called *py\_entitiymatching*.

Magellan combines lots of features like blockers, different matchers, debuggers and even external library support, however; it doesn’t have any GUI (Graphical User Interface). Users have to use Magellan via Python commands or Python scripts. In addition to that, Magellan requires manual labeling (*Figure 3*) for learning based matchers. Therefore, the usability of the Magellan isn’t the best. Nevertheless, it has great documentation that explains each operation in the system. It also has step-by-step examples for users that aren’t familiar with the Python language. Reading the documentation is a mandatory step to use the Magellan system.

Magellan isn’t an automated system. User has to do most of the tasks. For instance, manual labeling requires high concentration and takes a lot of time. That’s why users want to use blockers before the labeling step. However, finding the best blocker or even a combination of blockers takes extra time. After finding the best blocker/s and labeling, searching for the best matcher step begins. Like the previous steps, it takes time to find the best matcher or combinations of matchers. That’s why the user burden on the Magellan system is very high compared to some of the EM systems. Table 7 shows the Magellan system features.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GUI** | **Input** | **# Index methods** | **# Compare methods** | **Debugger** | **Profiling** | **Require-ments** | **Other** |
| No | CSV | 5  (Custom func. blocker) | 6 Learning  1 Rule based | Yes  (Blocker & matcher) | Yes | Python | Special down sampling algorithm |

Table 7: Magellan System Features

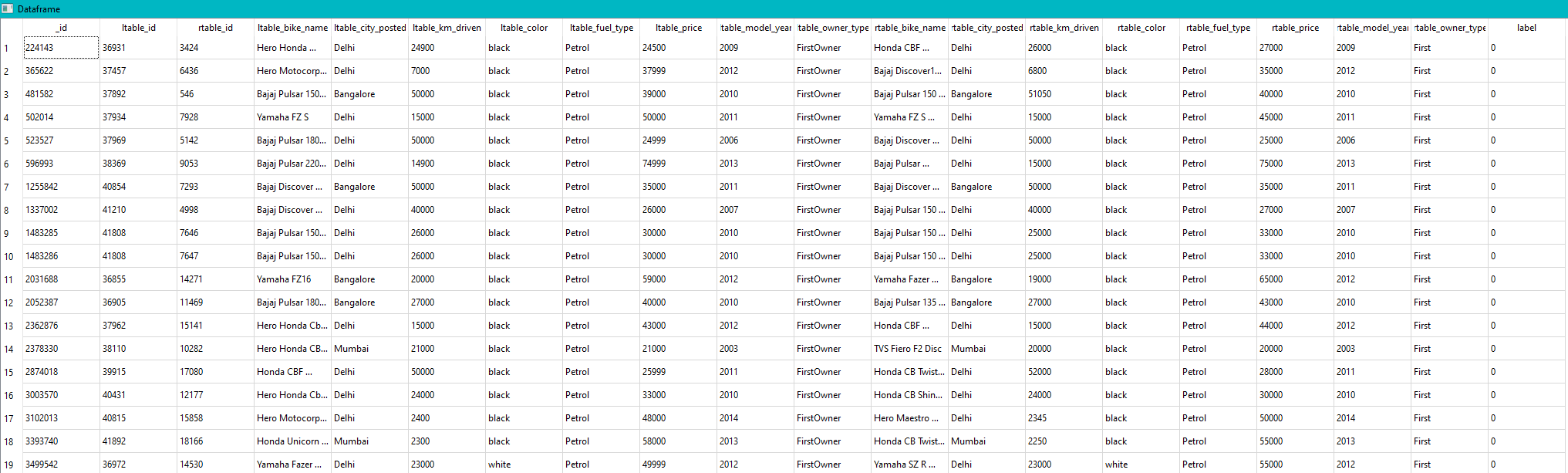


Figure 3: Magellan Manual Labeling Window

### SERF

SERF (Stanford Entity Resolution Framework) is a research-based EM framework developed by Stanford University. SERF is an EM program coded with Java language. That’s why it requires Java to run. However, JRE wouldn’t be sufficient to use the SERF because, users have to code their own Java classes for each dataset. Thus, JDK is required to use SERF.

SERF doesn’t have any GUI. Also, it lacks documentation, so users have to figure out how to use SERF. There are some research papers about SERF but, they only explain SERF’s matching algorithm not the system or how to use it. There are three different classes for three distance methods (Equal, Numeric distance and Jaro-Wrinkler distance). Users have to use these three matcher classes in their custom class for each attribute they want to use in the match detection. Also, there aren’t any weight values for attributes. SERF only checks the tuples’ attribute similarities. If the similarity scores are higher than the threshold, it checks the next attribute. If the score is lower than the threshold, it returns false for that tuple. After coding custom classes, users have to compile the SERF project to run it.

Original SERF system which located on the Stanford University website only takes XML file types as input data. However, there is an updated version in the GitHub that takes only CSV file types. The difference between these two SERF systems are some additional classes for custom datasets and CSV file converter. The GitHub version takes CSV files but users have to change CSVconverter class for their dataset attributes in order system to work. That’s why in this project, the original XML version used with CSV to XML convertor (Python script) in the experiments.

Using SERF for non-software people would be very hard. Because, users have to write Java code in order to use the system. Java language familiarity and knowledge is mandatory for SERF usage. Therefore, the usability of the SERF is very poor compared to other systems. Table 8 shows the SERF system features.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GUI** | **Input** | **# Index methods** | **# Compare methods** | **Debugger** | **Profiling** | **Requirem-ents** | **Other** |
| No | XML | None | 3  (Equal, Numeric, Jaro-Wrin.) | No | No | Java JDK | - |

Table 8: SERF System Features

### FRIL

FRIL (Fine-grained Records Integration and Linkage tool) is an open-source research-based entity matching system. It’s a Java-based program hence it requires JRE to run it. FRIL has good documentation that explains how to use the system. Nonetheless, it’s very easy to use thanks to its GUI. Users could easily understand the system and use it without even reading the documentation.

FRIL takes several different input file types such as CSV, Excel, text or even some databases. After importing data, users can profile it and see the dataset’s attribute features. FRIL consists of two different modes: Linkage and Deduplication. Linkage mode takes two different datasets as an input but deduplication only takes one. In both modes there is a middle step where users can choose matchers and blockers for dataset attributes. There are eight different matcher/distance methods and five indexing techniques. After choosing the matcher for one attribute, users can see its results on the selected column. So that, users can change the matcher’s feature according to debugger results. FRIL system features are shown in Table 9 and UI are shown in Figure 4-19.

|  |  |
| --- | --- |
| Matchers:   * Equal fields (Exact) * Edit distance * Numeric distance * Date distance * Q-gram distance * Soundex * Jaro-Wrinkler distance * Street-address distance | Indexing:   * Nested loop join (Standard, Non-blocker) * Blockers:   + Equal fields   + Soundex   + Prefix * Sorted neighbourhood * SVM join (experimental) |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GUI** | **Input** | **# Index methods** | **# Compare methods** | **Debugger** | **Profiling** | **Require-ments** | **Other** |
| Yes | CSV, Excel, Text, DB | 5  (Blocking, Sorted-Neigh.) | 8  (Edit dis., Q-gram, Soundex) | Yes | Yes | Java JRE | 2 modes (Linkage&Deduplication) |

Table 9: FRIL System Features

|  |  |
| --- | --- |
| Figure 4: FRIL Input Attribute Selection | Figure 5: FRIL Data Profiling |
| Figure 6: FRIL Deduplication Mode | Figure 7: FRIL Linkage Mode |
| Figure 8: FRIL Matcher/Distance Selection | Figure 9: FRIL Q-Gram Distance Properties |
| Figure 10: FRIL Date Distance Properties | Figure 11: FRIL Matcher/Distance Debugger |
| Figure 12: FRIL Blocker Selection | Figure 13: FRIL Sorted Neighbourhood Method |
| Figure 14: FRIL Manual Review Configuration | Figure 15: FRIL Automatic Attribute Weight Calculator |
| Figure 16: FRIL Result Saver | Figure 17: FRIL Matcher in Action |
| Figure 18: FRIL Matcher Confidence Score Chart | Figure 19: FRIL Result Window |

### Febrl

Febrl is a research-based entity matching program. Febrl is a Python program and it requires three different python libraries. These are PyGTK for GUI, libsvm for support vector machine matchers and matplotlib for graphs. It has great documentation that explains the system and how to use it.

Febrl has three different modes: Linkage, Deduplication and Standardisation. Linkage mode takes two datasets and other modes take one. Febrl can import CSV, text with different separator and SQL data. Standardisation separates columns by four prebuilt attribute types: date, phone, name and address.

Febrl’s explore (profiling) tab shows attribute features such as missing values, type and min-max. Febrl has seven different indexing methods. Some of them are blocking, Q-gram, sorting and suffix array. The next tab consists of comparison methods of the Febrl. There are twenty-six distinct distance methods. Age, date, edit-distance, Jaro, soundex, bag-distance and token-set are a few of them. Users can set agreeing weight, disagreeing weight, missing value weight and threshold values in this tab. The next step is choosing the classifier and Febrl has six of them. KMeans, SVM, FarthestFirst are some of the classification methods. Each method has four or five adjustable values. After choosing indexing, comparison and classification methods users can determine output files. Then the system starts to run matcher algorithms and shows the resulting graph in the evaluation tab. Before running the matcher, users can choose to save the created Python script which consists of all selected methods.

Febrl has the highest number of indexing and comparison/classification methods among other tested systems. Table 10 shows Febrl system features and Figure 20-27 shows Febrl UI windows.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GUI** | **Input** | **# Index methods** | **# Compare methods** | **Debugger** | **Profiling** | **Requirem-ents** | **Other** |
| Yes | CSV, Text, DB | 7  (Blocking, Q-Gram, Sorting) | 26  (Edit dis., Q-gram, Date) | No | Yes | Python,  PyGTK,  LibSVM,  Matplotlib | 3 modes (linkage,dedup., standr.) |

Table 10: Febrl System Features

|  |  |
| --- | --- |
| Figure 20: Febrl Deduplication Data Import | Figure 21: Febrl Linkage Data Import |
| Figure 22: Febrl Deduplication Data Profiling | Figure 23: Febrl Deduplication Indexing Methods |

|  |  |
| --- | --- |
| Figure 24: Febrl Deduplication Comparison Methods | Figure 25: Febrl Deduplication Classification Methods |
| Figure 26: Febrl Deduplication Output & Run | Figure 27: Febrl Match Evaluation Graph |

### Data Ladder

Data Ladder is one of the commercial systems. Thus, it has a lot of features that makes the system easy to use. It has great GUI, also shows each steps for entity matching operation in the left side in the GUI.

Data Ladder accepts more than one dataset simultaneously. That means users can compare multiple datasets. Magellan takes two tables while SERF takes only one table as an input. FRIL could take one or two depending on its modes. Data Ladder can take more than two datasets. Also, users can choose which tables to compare. Comparison could be only one table, between two tables or within two tables as shown in Figure 28.

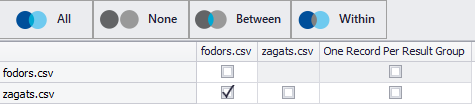


Figure 28: Data Ladder Match Table Selection

Data Ladder can import numerous types of data such as text with different separators, Excel, more than ten types of databases, CRM, social media, email… After importing data, users can use the data profiling tool to see dataset attribute features like emptiness, type and min-max values. The next step is Address Verification. Data Ladder provides this feature with GPS and zip code data. However, this project’s datasets don’t have any precise address attribute therefore, this feature couldn’t use in the experiments. After that, user can modify the dataset in the Data Cleansing & Standardization step. There are different features in this step. Some of them are change lower-uppercase, remove or change some characters in the columns and removing letters or digits. There are also some column types like first name, full name and address. These types can separate the values of the selected columns. For instance, while working on the Restaurants dataset, *addr* attribute selected as Address type. Data Ladder splits *addr* column and creates new columns such as Number, Building, Street, City… These new columns can be used in matching conditions, however; in this project instead of auto-generated columns initial *addr* attribute used.

The last step for matcher operation is Matcher configuration. In this step, users have to choose distant methods for tuple comparison. There are also threshold and weight values that can be adjustable. After that matcher finishes its operation, it shows detailed results of matched pairs or groups. Users could see similarity confidence in each match. Also, Data Ladder can create and printable PDF report of the matcher operation. This report consists of the number of matched tuples, confidence graph and runtime of each operation.

After matcher operation, users can merge matched tuples with selected attribute features. The final operation and step of the Data Ladder is Data Export. Users can export merged or matched datasets in this section of the system. There is also a scheduled task option for automatically runnable deduplication detection operation. Table 11 and Figure 29-40 show Data Ladder’s features and UI.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GUI** | **Input** | **# Index methods** | **# Compare methods** | **Debugger** | **Profiling** | **Require-ments** | **Other** |
| Yes | CSV, Excel, Text, DB, Social media | None | 4  (Fuzzy-match, phonetic) | No | Yes | None | Match report |

Table 11: Data Ladder System Features

|  |  |
| --- | --- |
| Figure 29: Data Ladder Window & Matcher Steps | Figure 30: Data Ladder Data Import Window |

|  |  |
| --- | --- |
| Figure 31: Data Ladder Importing Features | Figure 32: Data Ladder Data Profiling |
| Figure 33: Data Ladder Attribute Patterns | Figure 34: Data Ladder Profiling |
| Figure 35: Data Ladder Cleansing & Standardization | Figure 36: Data Ladder Matcher Selection |
| Figure 37: Data Ladder Matched Pairs | Figure 38: Data Ladder Matcher Similarity Results |
| Figure 39: Data Ladder Matcher Report | Figure 40: Data Ladder Matcher Durations |

### WinPure Clean & Match

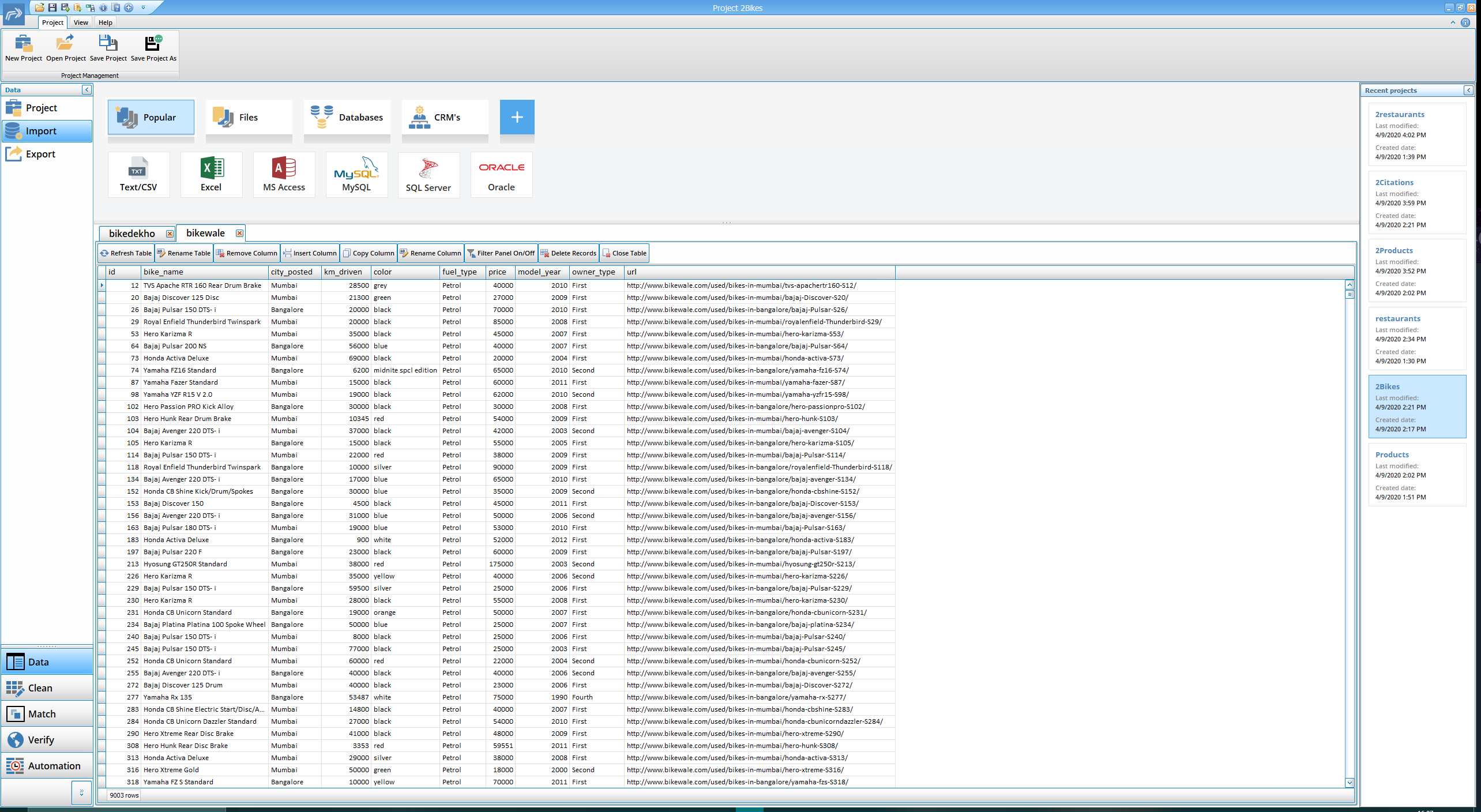
WinPure Clean & Match is another commercial data cleaning and entity matching system. It has GUI and its similar to the Data Ladder. Also, like in the Data Ladder example, WinPure could take more than two datasets for linkage. However, in demo mode, it is restricted with at most two datasets.

Data Ladder and WinPure are very similar in terms of both GUI and features. Like in Data Ladder, WinPure has the left side panel to show each step in the entity matching process. These steps are data import, data cleaning, match configuration and result report, merger and data export. Data importer can take text data with different separators like CSV, Excel data, seven different databases and two CRMs. In the cleaning step, the user can change or remove specific characters in the columns. Matcher of the WinPure has four distance methods: Fuzzy match, numeric match, exact match and date/time distance. Each attribute in the matcher has threshold and weight values but weight values can only be adjusted between 20 to 100. This was a problem in the products experiment because products have two attribute weights lower than 20 but WinPure doesn’t allow these (2%, 3%) values.

Some of the differences between WinPure and Data Ladder are max demo tuple numbers. Data Ladder could take up to 1M tuples data but WinPure only allows data with 10K tuples. Also, WinPure’s numeric distance method in the matcher doesn’t use a threshold value, so it only compares the number for an exact match. Thus, numeric range difference cannot be used in WinPure. Like in Data Ladder, WinPure has Address Verification feature as well but, in WinPure this feature is in experimental state. WinPure doesn’t have different table comparison modes like between, none or within in the matcher step. It only allows users to select which tables to use. Lastly, WinPure cannot separate some of the CSV files correctly. For instance, it couldn’t separate The Products dataset’s dimensions and weight attributes. These two columns had dirty data from the previous columns, so in the Products experiment simplified versions of both datasets used. Table 12 shows WinPure system features and Figure 41-47 show its UI elements.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GUI** | **Input** | **# Index methods** | **# Compare methods** | **Debugger** | **Profiling** | **Requirem-ents** | **Other** |
| Yes | CSV, Excel, Text, DB, CRM | None | 4  (Fuzzy, numeric, date) | No | Yes | None | - |

Table 12: WinPure System Features

Figure 41: WinPure Main Window

|  |  |
| --- | --- |
| Figure 42: WinPure Data Import Window | Figure 43: WinPure Importer Attribute Selection |
| Figure 44: WinPure Data Cleaning | |
| Figure 45: WinPure Data Profiling | |
| Figure 46: WinPure Match Configuration | Figure 47: WinPure Match Results |

### Management Ware Data Cleansing & Matching

Management Ware is another commercial data cleaning and entity matching system. It has a free demo version.

Management Ware system has GUI, however; it isn’t as easy as Data Ladder or WinPure. It takes time to understand how to use it. Data import types are text, CSV, Excel and three databases. There are both deduplication and match functionality. However, none of these methods have distance metrics. Both deduplication and matcher have condition and operation inputs. Condition option has AND and OR values. Operation has Equals, Starts with, Ends with and Contains/Like options. Management Ware cannot be compared with other systems because it doesn’t have distance metrics or weight values in matchers. Therefore, it didn’t have any experimental results in *Section 5*. Table 13 shows Management Ware features and Figure 48-53 show its UI.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GUI** | **Input** | **# Index methods** | **# Compare methods** | **Debugger** | **Profiling** | **Requireme-nts** | **Other** |
| Yes | CSV, Excel, Text, DB | None | None | No | Yes | None | - |

Table 13: Management Ware System Features

|  |  |
| --- | --- |
| Figure 48: Management Ware Main Window | Figure 49: Management Ware Data Import |
| Figure 50: Management Ware Attribute Selection | Figure 51: Management Ware Data Cleaning |

|  |  |
| --- | --- |
| Figure 52: Management Ware Deduplication | Figure 53: Management Ware Matcher & Merger |

## Who uses EM systems

There are different commercial entity matching systems that couldn’t be tested in this project due to license limitations. Most of the EM systems is part of the Data Quality Systems. These quality tools mostly used by information technology companies for data cleaning. Entity matching is part of the data cleaning operations.

Research-based systems generally used by researchers or students. Researchers mostly use other entity matching system to compare their new algorithm/system with these implemented systems. It is easier to use implemented systems rather than implementing each EM algorithms by yourselves.

# EXPERIMENTS

In this section tested systems experiment results will be shown. Some systems couldn’t be tested for runtime and accuracy performance. Systems tested are Magellan, SERF, FRIL, Data Ladder and WinPure. Some of them couldn’t run the Citations dataset and gave an error. Errors are shown as *“Error”* in runtime and accuracy tables.

Febrl could not be used because of its matcher methods. It has both comparison and classification techniques and it uses both of them to match tuples. Both the Restaurants and the Products datasets were tested with different combinations of the comparison and classification methods, but it only detected ‘*possible matches’* not real matches. Therefore, it didn’t generate any output files. Normally Febrl shows matcher precision, recall and F1 rates in the evaluation tab. However, since it couldn’t detect any matches it didn’t show these rates. Hence, there is no runtime or accuracy value for Febrl in the experiment result tables.

Management Ware doesn’t have any distance algorithms to use in matcher. It has only SQL database operations such as *like, contains* and *equals*. Thus, Management Ware couldn’t be used in the accuracy calculation.

## System Specs

This paper is a graduate-level course project’s report. Therefore, I used my laptop for all experiments.

Computer Specs:

* CPU: Intel I7 4710MQ @ 2.5 GHz (Boost: 3.5 GHz), 4 Cores – 8 Threads
* RAM: 16GB DDR3
* OS: Windows 10 Home (64 bit)
* HDD connection: SATA @6Gb/s

## Experiment Results

### Runtimes

Table 14 shows tested systems’ runtime values in (hh:mm:ss.ms) format. Magellan has two different runtime values in the table. The first one shows the Magellan debugging runtime. In debugging mode users trying to find the best blocker and matcher methods. That means applying different blockers/matchers to the dataset and debugging their results to improve them. There are lots of parameters that can be adjustable. That’s why debug mode takes a lot of time. However, after finding the best methods and parameters, users can put these into the Python script and run it. It takes less time to complete but it still requires manual labeling. That’s why Magellan takes a longer time to finish its operations. FRIL has also two rows because it has two modes and both of them were tested for all datasets.

These runtime scores show that commercial systems are way faster than the research-based systems. Also, Magellan’s manual labeling step increases its runtime significantly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Bikes** | **Restaurants** | **Citations** | **Products** |
| **Magellan**  **(Debug)** | 1:02:16.352 | 1:14:12.730 | Error | 0:45:50.581 |
| **Magellan**  **(Script)** | 0:06:29.107  (incl. labeling) | 0:01:50.758  (incl. labeling) | Error | 0:06:08.167  (incl. labeling) |
| **SERF** | 0:44:32.864 | 0:0:02.204 | Error | 0:53:37.298 |
| **FRIL**  **(Linkage)** | 0:02:25.935 | 0:0:02.988 | 05:20:25.722 | 0:0:05.849 |
| **FRIL (Deduplication)** | 0:02:47.507 | 0:0:0.181 | 02:47:14.080 | 0:0:01.573 |
| **Data Ladder** | 0:0:02.710 | 0:0:01.640 | 0:05:59.950  (2M tuples) | 0:0:04.910 |
| **WinPure** | 0:0:01.140 | 0:0:0.781 | 0:0:07.489  (20K tuples) | 0:0:01.203 |

Table 14: Dataset-System Runtime Table

### Accuracy

Most of the systems’ accuracy calculated with output file and matched tuple files from each dataset. Since the Bikes dataset doesn’t have any matched tuple file, this dataset couldn’t use in accuracy calculation. All systems’ except Magellan’s precision, recall and F1 rates calculated with system output files. Magellan doesn’t produce output files and it calculates each learning-based methods’ precision, recall and F1 rates automatically. Magellan accuracy results show these automatically calculated results. However, other systems accuracies calculated manually. For this reason, FRIL used to compare matched tuples data with output files of each experiment. FRIL comparison results show how many of the tuples in the output file are actually in the matched data. The only exception for using FRIL for match detection was SERF. SERF takes and produces XML files, therefore; FRIL couldn’t used for SERF accuracy calculation. Custom Python script developed for counting the correct matches in the SERF output by using the matched data of each dataset.

#### Accuracy of Magellan System

Magellan has five different blockers and six learning-based matchers. The first tables (*Table 13, 17, 21*) for each dataset show which blocker method used for the written attribute. The second tables (*Table 14, 18 ,22*) display matcher selection operation results. Magellan runs all six learning-based matchers with test data to find the best matcher. After this operation, users can choose the best one and use it for detecting the matches. However, in this project all six matchers are used. Third tables (*Table 15, 19, 23*) show two matchers’ debugger results. Only two of the matchers (Decision Tree & Random Forest) have debugger features thus only their debugger results presented in the tables. The last tables (*Table 16, 20, 24*) demonstrate all matchers final accuracy results. These matchers run on the input dataset and use manually labeled data as train data. The best scores painted in green.

##### Bikes Dataset in Magellan

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Blocker** | **Comparison Method** | **Comparison Features** |
| bike\_name | Overlap Blocker | Q-gram | Overlap: 1, Uses Words |
| price | Black Box Blocker | Numeric Distance | Range between: +10% or -10% |
| city\_posted | Attribute Equivalence Blocker | Equal Fields | - |
| color | Attribute Equivalence Blocker | Equal Fields | - |
| fuel\_type | Attribute Equivalence Blocker | Equal Fields | - |
| km\_driven | Black Box Blocker | Numeric Distance | Range between: +5% or -5% |
| model\_year | Attribute Equivalence Blocker | Equal Fields | - |
| owner\_type | Overlap Blocker | Q-gram | Overlap: 2, Q-gram: 2 |

Table 15: Magellan Bikes Dataset Blocker Features

* Labeled sample size: 100
* Name difference between 2 tables’ name attribute was *“Standard”* keyword. Bike Dekho dataset has this keyword but Bike Wale doesn’t have it.

|  |  |  |  |
| --- | --- | --- | --- |
| **Matcher** | **Average Precision** | **Average Recall** | **Average F1** |
| Decision Tree | 0.613333 | 0.513333 | 0.533333 |
| Random Forest | 0.600000 | 0.223333 | 0.306667 |
| Support Vector Machine | 0.400000 | 0.150000 | 0.213333 |
| Linear Regression | 0.320000 | 0.316667 | 0.244286 |
| Logistic Regression | 0.500000 | 0.283333 | 0.360000 |
| Naïve Bayes | 0.370000 | 0.490000 | 0.382143 |

Table 16: Bikes Matcher Selection Results

|  |  |  |
| --- | --- | --- |
| **Matcher Debugger** | **False Positives** | **False Negatives** |
| Decision Tree | 7/14 | 0/21 |
| Random Forest | 5/12 | 0/23 |

Table 17: Bikes DT & RF Debugger Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Matcher** | **Precision** | **Recall** | **F1** | **False Positives** | **False Negatives** |
| Decision Tree | 70.0% (7/10) | 77.78% (7/9) | 73.68% | 3 (out of 10 positive predictions) | 2 (out of 20 negative predictions) |
| Random Forest | 87.5% (7/8) | 77.78% (7/9) | 82.35% | 1 (out of 8 positive predictions) | 2 (out of 22 negative predictions) |
| Support Vector Machine | 66.67% (2/3) | 22.22% (2/9) | 33.33% | 1 (out of 3 positive predictions) | 7 (out of 27 negative predictions) |
| Linear Regression | 50.0% (5/10) | 55.56% (5/9) | 52.63% | 5 (out of 10 positive predictions) | 4 (out of 20 negative predictions) |
| Logistic Regression | 42.86% (3/7) | 33.33% (3/9) | 37.5% | 4 (out of 7 positive predictions) | 6 (out of 23 negative predictions) |
| Naïve Bayes | 66.67% (4/6) | 44.44% (4/9) | 53.33% | 2 (out of 6 positive predictions) | 5 (out of 24 negative predictions) |

Table 18: Bikes Every Matcher Evaluation Results

##### Restaurants Dataset in Magellan

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Blocker** | **Comparison Method** | **Comparison Features** |
| Name | Overlap Blocker | Q-gram | Overlap: 1, Uses Words, Ignores stop words (the, a, an) |
| Type | Attribute Equivalence Blocker | Equal Fields | - |

Table 19: Magellan Restaurants Dataset Blocker Features

* Labelled sample size: 50
* Both tables’ name attributes have lots of *“Café”* in them. This makes hard to block tuples with name attribute.

|  |  |  |  |
| --- | --- | --- | --- |
| **Matcher** | **Average Precision** | **Average Recall** | **Average F1** |
| Decision Tree | 0.4 | 0.4 | 0.4 |
| Random Forest | 0.4 | 0.4 | 0.4 |
| Support Vector Machine | 0.0 | 0.0 | 0.0 |
| Linear Regression | 0.2 | 0.2 | 0.2 |
| Logistic Regression | 0.4 | 0.4 | 0.4 |
| Naïve Bayes | 0.4 | 0.4 | 0.4 |

Table 20: Restaurants Matcher Selection Results

|  |  |  |
| --- | --- | --- |
| **Matcher Debugger** | **False Positives** | **False Negatives** |
| Decision Tree | 0/2 | 0/16 |
| Random Forest | 0/0 | 2/18 |

Table 21: Restaurants DT & RF Debugger Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Matcher** | **Precision** | **Recall** | **F1** | **False Positives** | **False Negatives** |
| Decision Tree | 100.0% (4/4) | 100.0% (4/4) | 100.0% | 0 (out of 4 positive predictions) | 0 (out of 11 negative predictions) |
| Random Forest | 100.0% (4/4) | 100.0% (4/4) | 100.0% | 0 (out of 4 positive predictions) | 0 (out of 11 negative predictions) |
| Support Vector Machine | 0.0% (0/0) | 0.0% (0/0) | 0.0% | 0 (out of 0 positive predictions) | 4 (out of 15 negative predictions) |
| Linear Regression | 100.0% (2/2) | 50.0% (2/4) | 66.67% | 0 (out of 2 positive predictions) | 2 (out of 13 negative predictions) |
| Logistic Regression | 100.0% (4/4) | 100.0% (4/4) | 100.0% | 0 (out of 4 positive predictions) | 0 (out of 11 negative predictions) |
| Naïve Bayes | 100.0% (4/4) | 100.0% (4/4) | 100.0% | 0 (out of 4 positive predictions) | 0 (out of 11 negative predictions) |

Table 22: Restaurants Every Matcher Evaluation Results

##### Citations Dataset in Magellan

Citations dataset didn’t work on the Magellan system. It stops working and gives below error message. This caused by non-clear data and column separation problem.

**Error Message:**

DtypeWarning: Columns (3,4) have mixed types. Specify dtype option on import or set low\_memory=False.

##### Products Dataset in Magellan

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Blocker** | **Comparison Method** | **Comparison Features** |
| Name | Overlap Blocker | Q-gram | Overlap: 1, Uses Words, Ignores stop words (the, a, an) |
| Type | Attribute Equivalence Blocker | Equal Fields | - |

Table 23: Magellan Products Dataset Blocker Features

|  |  |  |  |
| --- | --- | --- | --- |
| **Matcher** | **Average Precision** | **Average Recall** | **Average F1** |
| Decision Tree | 0.383333 | 0.500000 | 0.397143 |
| Random Forest | 0.700000 | 0.500000 | 0.526667 |
| Support Vector Machine | 0.333333 | 0.266667 | 0.260000 |
| Linear Regression | 0.243333 | 0.500000 | 0.321429 |
| Logistic Regression | 0.433333 | 0.400000 | 0.366667 |
| Naïve Bayes | 0.490000 | 0.633333 | 0.524286 |

Table 24: Products Matcher Selection Results

* Labelled sample size: 100
* Whilelabeling the Products samples, *ModelNo* and *Title* attributes were used. If both tables’ tuples have the *ModelNo* column, it makes labeling easier.
* *ModelNo* attribute was used in labeling, however; half of the tuples’ ModelNo columns are empty. Therefore, this column wasn’t used in blockers.
* Same items’ different colors labeled as match.

|  |  |  |
| --- | --- | --- |
| **Matcher Debugger** | **False Positives** | **False Negatives** |
| Decision Tree | 2/4 | 4/31 |
| Random Forest | 1/2 | 5/33 |

Table 25: Products DT & RF Debugger Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Matcher** | **Precision** | **Recall** | **F1** | **False Positives** | **False Negatives** |
| Decision Tree | 83.33% (5/6) | 45.45% (5/11) | 58.82% | 1 (out of 6 positive predictions) | 6 (out of 24 negative predictions) |
| Random Forest | 85.71% (6/7) | 54.55% (6/11) | 66.67% | 1 (out of 7 positive predictions) | 5 (out of 23 negative predictions) |
| Support Vector Machine | 80.0% (4/5) | 36.36% (4/11) | 50.0% | 1 (out of 5 positive predictions) | 7 (out of 25 negative predictions) |
| Linear Regression | 60.0% (6/10) | 54.55% (6/11) | 57.14% | 4 (out of 10 positive predictions) | 5 (out of 20 negative predictions) |
| Logistic Regression | 87.5% (7/8) | 63.64% (7/11) | 73.68% | 1 (out of 8 positive predictions) | 4 (out of 22 negative predictions) |
| Naïve Bayes | 45.45% (10/22) | 90.91% (10/11) | 60.61% | 12 (out of 22 positive predictions) | 1 (out of 8 negative predictions) |

Table 26: Products Every Matcher Evaluation Results

#### Accuracy of SERF

SERF output file merges matched tuples. So, some of the table cells have more than one value. It merges all matched tuples and creates groups. For instance, a grouped tuple’s id attribute can have ‘1, 5, 89, 245, 295’ values. That means SERF detected these five tuples as matched.

##### Bikes Dataset in SERF

* Number of matched tuples: 5505
* Number of groups: 945
* Number of groups that have more than 2 tuples (Incorrect matches): 291
* Number of groups that have 2 tuples (Correct matches, pairs): 654

##### Restaurants Dataset in SERF

|  |  |  |
| --- | --- | --- |
|  | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 51 | FN: 71 |
| **Negative Class** | FP: 241 | TN: 521 |

Table 27: Restaurants Dataset SERF Confusion Matrix

|  |  |  |
| --- | --- | --- |
| **Precision** | **Recall** | **F1** |
| 0.148688 | 0.455357 | 0.224176 |

Table 28: Restaurants Dataset SERF Accuracies

##### Citations Dataset in SERF

In first tries Java required more Heap Memory. So, SERF run with 12GB of heap memory for the Citations dataset. After 3 hours of running, SERF gave an error. The error was JRE *EXCEPTION\_ACCESS\_VIOLATION*.

##### Products Dataset in SERF

SERF product dataset result couldn’t detect any matches. Thus, it couldn’t merge tuples and number of tuples remained the same. Java console message was:

*Running RSwoosh on 24628 records.*

*After running RSwoosh, there are 24628 records.*

*Runtime(ms): 3217298*

|  |  |  |
| --- | --- | --- |
|  | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 0 | FN: 1154 |
| **Negative Class** | FP: 0 | TN: 23474 |

Table 29: Products Dataset SERF Confusion Matrix

|  |  |  |
| --- | --- | --- |
| **Precision** | **Recall** | **F1** |
| 0.0 | 0.0 | 0.0 |

Table 30: Products Dataset SERF Accuracies

#### Accuracy of FRIL

FRIL’s standard max Java heap memory limit is 600MB. However, Citations dataset required higher memory, thus memory limit increased only for Citations experiment.

##### Bikes Dataset in FRIL

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Weight** | **Comparison Method** | **Comparison Features** |
| bike\_name | 20 | Edit Distance | Approve: 0.2, Disapprove: 0.3 |
| price | 15 | Numeric Distance | Range between: +10% or -10% |
| city\_posted | 12 | Equal Fields | - |
| color | 12 | Equal Fields | - |
| fuel\_type | 12 | Equal Fields | - |
| km\_driven | 12 | Numeric Distance | Range between: +5% or -5% |
| model\_year | 12 | Equal Fields | - |
| owner\_type | 5 | Soundex Distance (5) | Approve: 0.2, Disapprove: 0.4 |

Table 31: Bikes Dataset FRIL Matcher Features

* Acceptance Level: 90/100
* Join method: Blocking search method on color attribute with Equal Fields method
* Linkage mode result: 18 linked matches
* Deduplication mode result: 482 duplicated tuples

##### Restaurants Dataset in FRIL

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Weight** | **Comparison Method** | **Comparison Features** |
| Name | 20 | Edit Distance | Approve: 0.2, Disapprove: 0.3 |
| City | 20 | Edit Distance | Approve: 0.1, Disapprove: 0.2 |
| Address | 20 | Street Address Distance | Approve: 0.2, Disapprove: 0.3 |
| Phone | 20 | Edit Distance | Approve: 0.2, Disapprove: 0.3 |
| Type | 20 | Equal Fields | - |

Table 32: Restaurants Dataset FRIL Matcher Features

* Acceptance Level: 100/100
* Join method: Blocking search method on Type attribute with Equal Fields method

|  |  |  |
| --- | --- | --- |
| ***LINKAGE*** | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 6 | FN: 106 |
| **Negative Class** | FP: 0 | TN: 858 |

Table 33: Restaurants Dataset FRIL Linkage Confusion Matrix

|  |  |  |
| --- | --- | --- |
| ***DEDUPLICATION*** | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 6 | FN: 106 |
| **Negative Class** | FP: 0 | TN: 858 |

Table 34: Restaurants Dataset FRIL Deduplication Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1** |
| **Linkage** | 1.0 | 0.053571 | 0.101694 |
| **Deduplication** | 1.0 | 0.053571 | 0.101694 |

Table 35: Restaurants Dataset FRIL Accuracies

##### Citations Dataset in FRIL

FRIL standard Java heap memory is 600MB but, for citations deduplication mode experiment this number wasn’t enough. Thus, firstly it set to 1GB then to 2GB.

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Weight** | **Comparison Method** | **Comparison Features** |
| Authors | 30 | Edit Distance | Approve: 0.2, Disapprove: 0.4 |
| Title | 70 | Edit Distance | Approve: 0.2, Disapprove: 0.4 |

Table 36: Citations Dataset FRIL Matcher Features

* Acceptance Level: 100/100
* Join method: Prefix of blocking attribute (Title – Edit Distance). Prefix length: 4

|  |  |  |
| --- | --- | --- |
| ***LINKAGE*** | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 33920 | FN: 523631 |
| **Negative Class** | FP: 1236 | TN: 4301749 |

Table 37: Citations Dataset FRIL Linkage Confusion Matrix

|  |  |  |
| --- | --- | --- |
| ***DEDUPLICATION*** | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 50942 | FN: 507845 |
| **Negative Class** | FP: 2369 | TN: 4283594 |

Table 38: Citations Dataset FRIL Deduplication Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1** |
| **Linkage** | 0.964842 | 0.060702 | 0.114218 |
| **Deduplication** | 0.955562 | 0.091165 | 0.166449 |

Table 39: Citations Dataset FRIL Accuracies

##### Products Dataset in FRIL

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Weight** | **Comparison Method** | **Comparison Features** |
| Brand | 25 | Edit Distance | Approve: 0.2, Disapprove: 0.3 |
| Title | 25 | Edit Distance | Approve: 0.2, Disapprove: 0.4 |
| Price | 35 | Numeric Distance | Range between: +10% or -10% |
| Model No | 10 | Edit Distance | Approve: 0.2, Disapprove: 0.3 |
| Dimensions | 3 | Edit Distance | Approve: 0.2, Disapprove: 0.4 |
| Weight | 2 | Edit Distance | Approve: 0.2, Disapprove: 0.4 |

Table 40: Products Dataset FRIL Matcher Features

* Acceptance Level: 85/100
* Join method: Blocking search method on Brand attribute with Soundex blocker method

|  |  |  |
| --- | --- | --- |
| ***LINKAGE*** | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 24 | FN: 1119 |
| **Negative Class** | FP: 11 | TN: 24604 |

Table 41: Products Dataset FRIL Linkage Confusion Matrix

|  |  |  |
| --- | --- | --- |
| ***DEDUPLICATION*** | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 7 | FN: 1147 |
| **Negative Class** | FP: 1566 | TN: 23055 |

Table 42: Products Dataset FRIL Deduplication Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1** |
| **Linkage** | 0.685714 | 0.020797 | 0.040369 |
| **Deduplication** | 0.004450 | 0.006065 | 0.005133 |

Table 43: Products Dataset FRIL Accuracies

#### Accuracy of Data Ladder

Data Ladder tested with the same datasets. These datasets attribute weights and thresholds were mentioned in *Section 2*. Some datasets have the apostrophe characters at the beginning and the end of the values. In this case these characters removed within Data Cleansing & Standardization module. For instance, in the Restaurant dataset:

* Removed apostrophes (‘) from all columns
* Removed (/-) characters from the phone column. After this, phone columns have only digits.

While using the Data Ladder:

* Restaurants and Products datasets used as one table/file
* Restaurants, Bikes and Citations datasets used as two tables/files
* Citations dataset tables have more than a million tuples. However, Data Ladder’s free trial demo only accepts one million tuples. Therefore, in the Citations datasets experiment, Data Ladder only compared the first 1M tuples in both datasets.

##### Restaurants Dataset in Data Ladder

Restaurants dataset tested in both linkage (2 tables) and deduplication (1 table) modes in Data Ladder. Both modes results were the same.

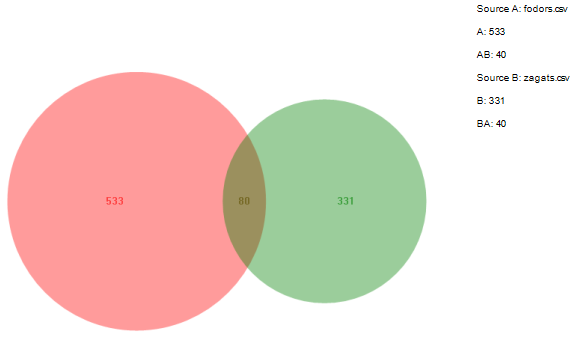


Figure 54: Restaurants Dataset Data Ladder Report

|  |  |  |
| --- | --- | --- |
|  | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 39 | FN: 72 |
| **Negative Class** | FP: 1 | TN: 824 |

Table 44: Restaurants Dataset Data Ladder Confusion Matrix

|  |  |  |
| --- | --- | --- |
| **Precision** | **Recall** | **F1** |
| 0.975 | 0.348214 | 0.513157 |

Table 45: Restaurants Dataset Data Ladder Accuracies

##### Citations Dataset in Data Ladder

Citations dataset didn’t get all of the tables. Citeseer dataset has 2.3M and the DBLP dataset has 1.8M tuples but Data Ladder’s demo version didn’t allow users to import more than 1 million tuples datasets. Thus, it only imported the first million tuples of both datasets. Matcher results for these two missing datasets were 0. Data Ladder couldn’t find any matches in these datasets. Therefore, its precision, recall and F1 rates are 0.

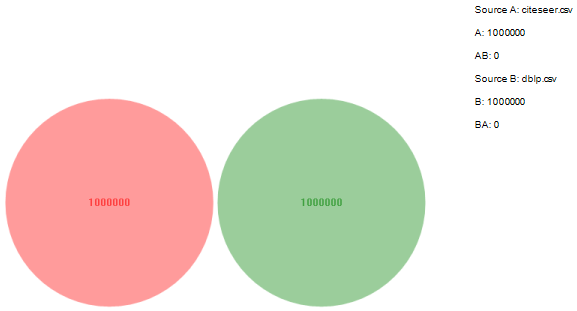


Figure 55: Citations Dataset Data Ladder Match Report

|  |  |  |
| --- | --- | --- |
| **Precision** | **Recall** | **F1** |
| 0.0 | 0.0 | 0.0 |

Table 46: Citations Dataset Data Ladder Accuracies

##### Products Dataset in Data Ladder

Products tested in two modes as well. However, in linkage mode Data Ladder couldn’t detect any matches. But, in deduplication mode (with both amazon and Walmart data) it found more than 1000 matches.

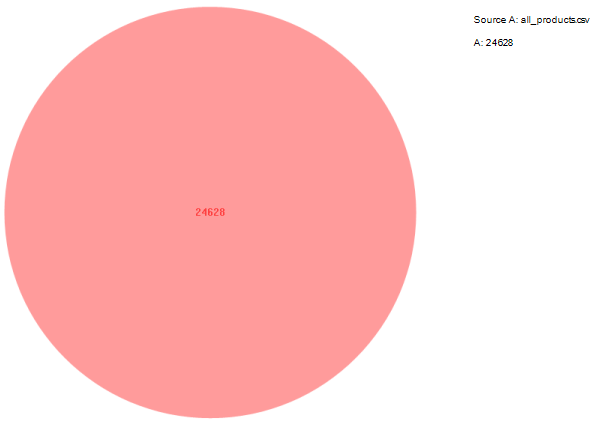


Figure 56: Products Dataset Data Ladder Match Report

|  |  |  |
| --- | --- | --- |
|  | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 1146 | FN: 8 |
| **Negative Class** | FP: 526 | TN: 23482 |

Table 47: Products Dataset Data Ladder Confusion Matrix

|  |  |  |
| --- | --- | --- |
| **Precision** | **Recall** | **F1** |
| 0.813342 | 0.993067 | 0.894263 |

Table 48: Products Dataset Data Ladder Accuracies

#### Accuracy of WinPure

##### Bikes Dataset in WinPure

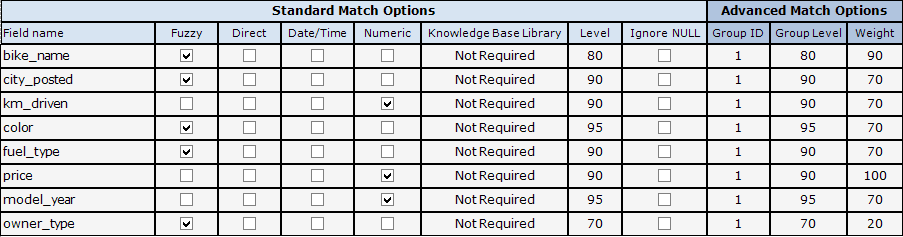
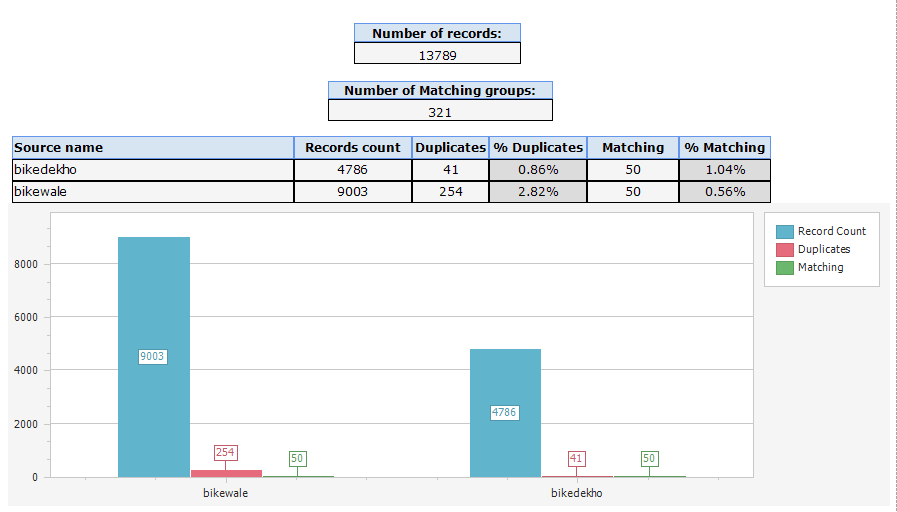


Figure 57: WinPure Bikes Dataset Match Configuration

* Dekho Matches: 50
* Wale Matches: 50
* Dekho Deduplicates: 254
* Wale Deduplicates: 41



Bikes dataset doesn’t have matched tuple csv file, therefore; it doesn’t have accuracy tables.

##### Restaurants Dataset in WinPure

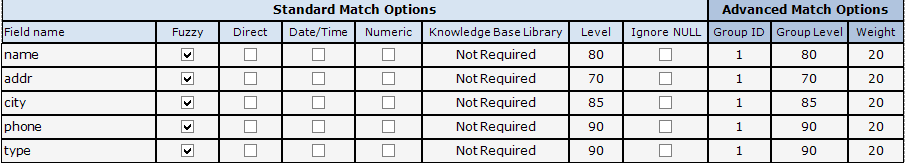


Figure 58: WinPure Restaurants Dataset Match Configuration

* Fodors Matches: 47
* Zagats Matches: 47
* Fodors Deduplicates: 1
* Zagats Deduplicates: 2

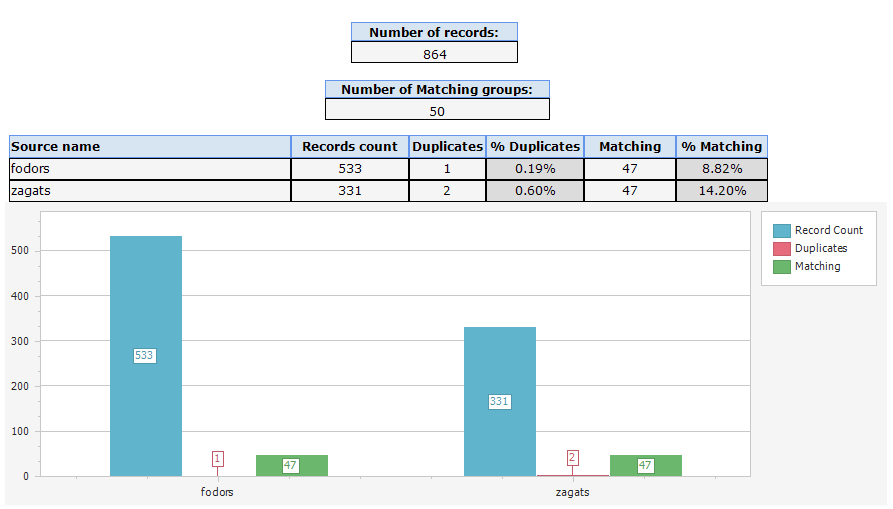


Figure 59: WinPure Restaurants Dataset Match Report

|  |  |  |
| --- | --- | --- |
|  | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 47 | FN: 72 |
| **Negative Class** | FP: 3 | TN: 814 |

Table 49: Restaurants Dataset WinPure Confusion Matrix

|  |  |  |
| --- | --- | --- |
| **Precision** | **Recall** | **F1** |
| 0.94 | 0.419642 | 0.580246 |

Table 50: Restaurants Dataset WinPure Accuracies

##### Citations Dataset in WinPure

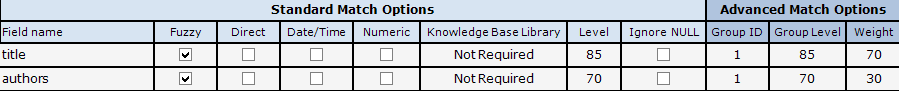


Figure 60: WinPure Citations Dataset Match Configuration

WinPure demo version only imported the first 10k tuples of both datasets. Therefore, the matched tuple number is lower. Citations matched tuples dataset has whole Citeseer and DBLP data matches, 4336905 use as a complete dataset size (not 20000).

* Citeseet Matches: 417
* DBLP Matches: 460
* Citeseet Deduplicates: 575
* DBLP Deduplicates: 757

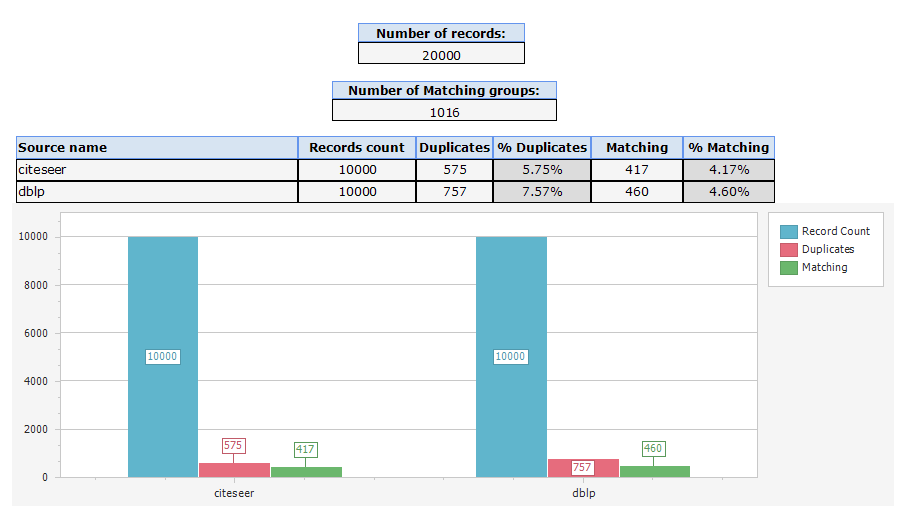


Figure 61: WinPure Citations Dataset Match Report

|  |  |  |
| --- | --- | --- |
|  | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 877 | FN: 557910 |
| **Negative Class** | FP: 1332 | TN: 4334696 |

Table 51: Citations Dataset WinPure Confusion Matrix

|  |  |  |
| --- | --- | --- |
| **Precision** | **Recall** | **F1** |
| 0.397012 | 0.001569 | 0.003126 |

Table 52: Citations Dataset WinPure Accuracies

##### Products Dataset in WinPure

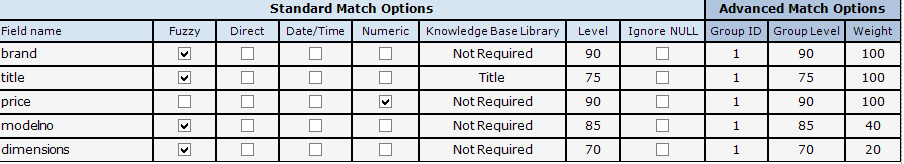


Figure 62: WinPure Products Dataset Match Configuration

WinPure demo version only imported the first 10k tuples of the amazon dataset. Therefore, the matched tuple number is lower. Since products matched tuples dataset has whole Amazon and Walmart data matches, 24628 use as a complete dataset size (not 12254). Also, WinPure is very bad at handling CSV data. In the Products dataset test, it couldn’t separate some of the columns in Walmart data. Thus, simplified Amazon and Walmart CSV files used in the WinPure experiment. Simplified datasets only have seven attributes used in the matcher (id, brand, title, price, modelno, dimensions, weight). Weight attribute couldn’t have used in this test because WinPure detected Amazon’s weight attribute type as string and Walmart’s as an integer. Hence, it didn’t allow a string to integer comparison in both fuzzy match and numeric match.

* Amazon Matches: 36
* Walmart Matches: 26
* Amazon Deduplicates: 729
* Walmart Deduplicates: 97

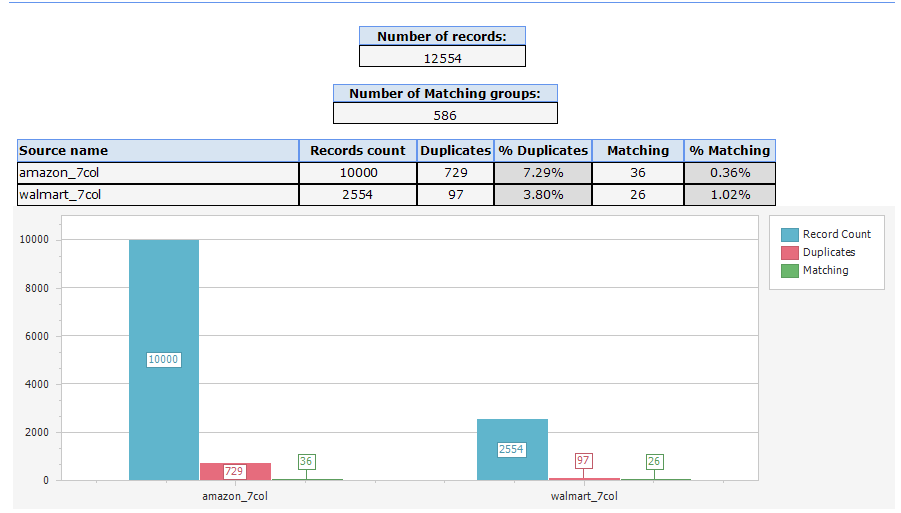


Figure 63: WinPure Products Dataset Match Report

|  |  |  |
| --- | --- | --- |
|  | **Positive Prediction** | **Negative Prediction** |
| **Positive Class** | TP: 62 | FN: 992 |
| **Negative Class** | FP: 826 | TN: 23740 |

Table 53: Products Dataset WinPure Confusion Matrix

|  |  |  |
| --- | --- | --- |
| **Precision** | **Recall** | **F1** |
| 0.069820 | 0.058824 | 0.063852 |

Table 54: Products Dataset WinPure Accuracies

#### Accuracy of All Systems

In this section each datasets precision, recall and F1 scores shown for every system tested. Bikes dataset only has Magellan scores because, this dataset doesn’t have any matched tables. So, 3 accuracy scores could have calculated for each system other than Magellan. Magellan has the accuracy scores because it calculates these scores automatically. However, other systems precision, recall and recall rates calculated manually from the output of the systems.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Precision* | **Bikes** | **Restaurants** | **Citations** | **Products** |
| **Magellan** | 0.875 | 1.0 | Error | 0.875 |
| **SERF** | N/A | 0. 148688 | Error | 0.0 |
| **FRIL (Linkage)** | N/A | 1.0 | 0.964842 | 0.68571 |
| **FRIL (Deduplication)** | N/A | 1.0 | 0.955562 | 0.004450 |
| **Data Ladder** | N/A | 0.975 | 0.0 | 0.813342 |
| **WinPure** | N/A | 0. 94 | 0. 397012 | 0. 069820 |

Table 55: Dataset-System Precision Score Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Recall* | **Bikes** | **Restaurants** | **Citations** | **Products** |
| **Magellan** | 0.7778 | 1.0 | Error | 0.6364 |
| **SERF** | N/A | 0. 455357 | Error | 0.0 |
| **FRIL (Linkage)** | N/A | 0.05357 | 0.060702 | 0.0208 |
| **FRIL (Deduplication)** | N/A | 0.05357 | 0.091165 | 0.006065 |
| **Data Ladder** | N/A | 0.348214 | 0.0 | 0.993067 |
| **WinPure** | N/A | 0. 419642 | 0. 001569 | 0. 058824 |

Table 56: Dataset-System Recall Score Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *F1* | **Bikes** | **Restaurants** | **Citations** | **Products** |
| **Magellan** | 0.8235 | 1.0 | Error | 0.7368 |
| **SERF** | N/A | 0. 224176 | Error | 0.0 |
| **FRIL (Linkage)** | N/A | 0.10169 | 0.114218 | 0.04037 |
| **FRIL (Deduplication)** | N/A | 0.10169 | 0.166449 | 0.005133 |
| **Data Ladder** | N/A | 0.513157 | 0.0 | 0.894263 |
| **WinPure** | N/A | 0. 580246 | 0. 003126 | 0. 063852 |

Table 57: Dataset-System F1 Score Table

## Evaluation

Accuracy scores of all tested systems were compared with 3 different rates: precision, recall and F1. The following equations show how these rates are calculated.

* **Precision =**
* **Matcher Precision =**
* **Recall =**
* **Matcher Recall =**
* **F1 =**

### Precision

Precision rate shows the detected matches correctness. If the detected tuple pairs represent the same real-world entities, they will be count as correct matches and increase the precision score.

The precision graph shows that all systems except SERF perform near perfect with restaurants dataset. For the Citation dataset, Magellan and SERF gave and error and couldn’t process the dataset. On the other hand, Data Ladder took only first 1M tuples of both Citeseer and DBLP datasets but it couldn’t find any matches. FRIL’s performance on the Citations dataset is over 95% but WinPure couldn’t perform well with 40% precision rate. Products datasets scores are mixed. Firstly, SERF couldn’t detect any matches within products datasets. SERF merges matched tuples but after the matcher operation it didn’t merge any tuples. FRIL deduplication mode and WinPure didn’t perform well for the Products dataset and their rates are less than 7%. Magellan, FRIL linkage and Data Ladder precision rates on the Products are very high compared to other 3 systems. FRIL linkage mode has the best precision scores across all datasets. Also, Magellan and Data Ladder have better rates on 2 datasets but both of them couldn’t work on the Citations datasets.

### Recall

Recall rate indicates the detection rate of the matches. If the recall rate is 1, that means the system was found all the same real-world entities in the dataset. Lower recall rate means a lower match detection rate.

Recall rates are way lower compared to precision rates. That means all systems can detect the correct matched tuples but they cannot detect all of the matches. Magellan has higher scores in recall rates as well because it automatically calculates these three rates. It doesn’t produce the output file so three rates couldn’t be calculated with all datasets matches files. If we can calculate the real rates of the Magellan, it would be lower than 1.

Restaurants dataset has higher recall rates than the other two datasets because it only has 112 matches while others have more than a thousand matches. FRIL linkage mode had the best precision rate but both linkage and deduplication modes recall performance are less than 10%. SERF has the best restaurants dataset rate with the detection of nearly half of the matches. Commercial systems’ the Restaurants dataset scores are lower than the SERF and Magellan but they performed better than the FRIL. For the Citations dataset, only three of the systems worked and their performance was very low. This dataset has 560K matches and the best system (FRIL deduplication) could only detect 5K (10%) of them. Productions dataset has empty values and some systems could read the datasets correctly. However, Data Ladder performs near 100% on this dataset. Nevertheless, other commercial system WinPure couldn’t compete with neither Data Ladder nor Magellan. Magellan and Data Ladder are the best systems in terms of detecting all matches.

### F1

F1 indicates both precision and recall rates performance. It is similar to the average of these 2 rates.

Magellan has the highest F1 scores because of its recall rates. The second best system is Data Ladder thanks to its Products recall rate. FRIL had the best precision rates, however; its recall rates are very low so it has lower F1 rates compared to other systems. WinPure performed well on restaurants dataset thus it has higher F1 rates than the Data Ladder. SERF only worked correctly on restaurants dataset but it could only detect half of the matches. These F1 scores prove that no system is perfect for every dataset. Accuracy of the results depending on dataset and selection of the best distance algorithms in matchers. Some of the systems don’t have much variety of matches, therefore; selecting the most suitable one for the dataset isn’t possible. The more distance algorithms system has, the more suitable could be found for each distinct attribute.

# CONCLUSION

Entity Matching is still one of the top topics in data management research area. However, lack of implemented algorithms and systems is a problem. New entity matching algorithms should have open-source implementations so that other researchers could test them. This will increase the number of comparisons and survey papers in EM research area. In this project, commercial entity matching systems compared to research-based implemented algorithms in terms of result accuracy, runtime performance, system usability and additional features. Experiment results show that none of the systems are perfect in terms of those four metrics. Commercial systems have higher usability and better runtime performance compared to research-based systems. However, research-based systems may have better additional features such as external library support, and some of the research-based algorithms outperform commercial systems for result accuracy.

# REFERENCES

1. Barateiro, José, and Helena Galhardas. "A survey of data quality tools." Datenbank-Spektrum 14.15-21 (2005): 48.
2. Benjelloun, Omar, et al. "Swoosh: a generic approach to entity resolution." The VLDB Journal 18.1 (2009): 255-276.
3. Bharambe, Dewendra, Susheel Jain, and Anurag Jain. "A survey: detection of duplicate record." International Journal of Emerging Technology and Advanced Engineering 2.11 (2012): 298-307.
4. Christen, Peter. "Febrl- an open source data cleaning, deduplication and record linkage system with a graphical user interface." Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008.
5. Christen, Peter. "A survey of indexing techniques for scalable record linkage and deduplication." IEEE transactions on knowledge and data engineering 24.9 (2011): 1537-1555.
6. Elmagarmid, Ahmed K., Panagiotis G. Ipeirotis, and Vassilios S. Verykios. "Duplicate record detection: A survey." IEEE Transactions on knowledge and data engineering 19.1 (2006): 1-16.
7. Jurczyk, Pawel, et al. "FRIL: a tool for comparative record linkage." AMIA annual symposium proceedings. Vol. 2008. American Medical Informatics Association, 2008.
8. Konda, Pradap, et al. "Magellan: Toward building entity matching management systems." Proceedings of the VLDB Endowment 9.12 (2016): 1197-1208.
9. Köpcke, Hanna, and Erhard Rahm. "Frameworks for entity matching: A comparison." Data & Knowledge Engineering 69.2 (2010): 197-210.
10. Patil, Aparna Ajit, and Dhanashree Kulkarni. "A survey on: secure data deduplication on hybrid cloud storage architecture." International Journal of Computer Applications 110.3 (2015).