Master Data Management using Record Linkage Toolkit for Integrating Lecturer Master Data

M. Miftakul, Ami, Adi Sutrisman, and Yevi Dwitayanti

1 Department of Computer Engineering, Politeknik Negeri Sriwijaya, Palembang 30139, Indonesia
2 Department of Accounting, Politeknik Negeri Sriwijaya, Palembang 30139, Indonesia

Abstract. Merging databases from different data sources is one of the important tasks in the data integration process. This study will integrate lecturer data from data sources in the application of academic information systems and research information systems at the Sriwijaya State Polytechnic. This integration of lecturer data will later be used as a single data as master data that can be used by other applications. Lecturer data in the academic section contains 444 records, while those from the p3m section contain 443 records. An important task in the database merging process is to eliminate duplicate records. One of the important libraries in the formation of this master data management uses the record linkage toolkit which is implemented in the python programming language. The steps taken are preprocessing, generating candidate record pairs, compare pairs, score pairs, and finally the data link to merge the two data sources. In this study, 5 fields, namely username, name, place of birth, date of birth, and gender, from each data source were used to measure the level of record similarity. The result of this research is the formation of lecturer master data from the merging of the two sources.

1 Introduction

In the context of master data management (MDM), one or more records in a relational database concept can refer to an entity. The formation of master data management involves several tasks, including entity resolution, entity matching, and relation extraction[1].

The data that is processed in the formation of the master data management can come from the same data source or be heterogeneous[2]. Heterogeneous data is also related to the concept of data interoperability, where data can come from various sources[3] and involves web services technology or web application programming interface (API)[4].

One of the main tasks in establishing a master data management is to eliminate duplication of data in the source database[5]. Record linkage can be understood as a process for extracting records from various data sources and combining them to form a single entity[6], for both structured and unstructured data[7]. The same thing was also conveyed by[8] which defines that record linkage is a step to identify a number of records that refer to the same thing. In many cases, these records refer...
According to record linkage is used to improve data quality and integration at the data mining and data warehouse stages and decision-making processes. Even according to The record linkage stage is part of the cleaning stage in data mining. Record linkage itself is usually implemented into datasets that do not have a unique identifier. The term record linkage according to has several equivalents, namely de-duplication and entity resolution. Other terms were also conveyed by as data matching, probabilistic matching, and instance identification.

Various models of completion of record linkage have been studied, including using the Bayesian approach as an unsupervised approach. This study aims to detect duplication in the database under investigation. The most common approach in implementing record linkage is to use string similarity. According to that the string similarity level \( s \), \( s = \text{sim}(a_i, a_j) \) is a similarity function that calculates the level of similarity between two attribute values \( a_i \) and \( a_j \), which is in the range of values \( 0 \leq s \leq 1 \).

- \( \text{sim}(a_i, a_i) = 1 \), is the result of a string similarity comparison where the two strings are exactly the same value.
- \( \text{sim}(a_i, a_j) = 0 \), is a result of calculating string similarity where the two strings are values that are different from one another.
- \( 0 < \text{sim}(a_i, a_j) < 1 \), is the result of calculations between values that are exactly the same and different from one another. The greater the value, the more precise the degree of similarity will be.

In this study, a process for merging two datasets will be carried out containing information on lecturer data. The two datasets come from academic information systems and research information systems. The number of records in academic data is 444 records, while in Center for research and community service (P3M) data there are 443 records. The two databases will then be used as a single piece of data as master data which can later be used by other applications as lecturer master data.

2 Methods

Figure 1 shows the steps in merging databases using the record linkage approach. The steps taken include pre-processing, generating pairs, compare pairs, score pairs, and finally link data. All these stages are carried out using the record linkage toolkit library which is implemented in the python programming language.

![Fig. 1. Record Linkage Steps](https://example.com/fig1.png)

Some of these stages can be explained as follows:

- **Pre-processing**: At this stage, several standardizations of the dataset are carried out, such as changing the string to lowercase, eliminating more than one white space, uniforming,
3 Results and Discussion

3.1 Pre-Processing Step

The pre-processing stage is the most important stage. At this stage, uniform formatting and field selection are carried out for further processing. Not all fields in both data sources will be used. Only 5 pieces of field information will be used, namely username, name, place of birth, date of birth, and gender. At this stage, it is done to change the string in the name and place of birth fields to lowercase. In addition, if there is more than one white space, it will be deleted and made into one white space only. The following program code segments are used to perform pre-processing stages which are implemented on both dataset sources.

```python
# pre-processing steps

def casefolding(text):
    text = text.lower()
    text = text.replace(' s+', ' ')
    text = text.strip()
    return text
```

3.2 Generate Pairs

At this stage of generating pairs, indexing and blocking processes will be carried out. This stage will reduce the number of candidate record pairs, because not all records will be compared. Only records that have a high degree of similarity will be used for further processing. Without using blocking, for example, the number of records is 500, a pair of 500 x 500 record pairs or 250,000 will be generated. Of course, this will make computing more complex. By using indexing and blocking, the number of records can be reduced. Using the recordlinkage.Index() function will produce indexing objects obtained from 2 datasets, in this case dfA which comes from academic data, and dfB which comes from research and community service data.

```python
# Indexation step
indexer = rl.Index()
indexer.block(username)
candidate_links = indexer.index(dfA, dfB)
```

The important notation used to generate blocked pairs for a column is to call the block() function. After the indexer object is created, the indexer.index() method will be executed on
the two processed dataframes. Generate pairs will generate candidate_links containing multiindex information from both datasets.

3.3 Compare Pairs

At this stage a number of fields will be compared the level of similarity using the Jaro-Winkler similarity function. This method produces a range of values between 0 to 1. A value of 0 means that the two strings are not at all similar, while a value of 1 means that the two strings are the same value. The larger the resulting value, the more similar the two strings are. The following is a segment of program code to compare records.

```python
# Comparison step
compare_cl = rl.Compare()
compare_cl.string(username, username, method=jarowinkler, label=username)
compare_cl.string(name, name, method=jarowinkler, label=name)
compare_cl.string(pob, pob, method=jarowinkler, label=pob)
compare_cl.string(dob, dob, method=jarowinkler, label=dob)
compare_cl.string(sex, sex, method=jarowinkler, label=sex)
features = compare_cl.compute(candidate_links, dfA, dfB)
```

3.4 Score Pairs

By providing a threshold value (Θ) of more than (> 3), a set of records will be obtained as a result of matching. The following is a code segment for the score pairs program.

```python
# Classification step
matches = features[features.sum(axis=1) > 3]
```

3.5 Link Data

The data link is the final stage in a series of record linkage processes. This stage is tasked with merging or joining two datasets. The first step is to carry out the process to find duplicate records. After that, the duplicated data will be removed, and the results will be added in a new dataset as a result of the merger. The following is a segment of program code to perform the merging process.

```python
# get duplicate values
duplicate_rows = matches.index.get_level_values(1)
# Finding duplicates in data2
data1_duplicates = dfB[dfB.index.isin(duplicate_rows)]
# Finding new rows in data2
data2_new = dfB[~dfB.index.isin(duplicate_rows)]
# link both dataframes
data=dfA.append(data2_new)
# save the result of merging data
data.to_csv('output/integrated_data.csv')
```

Figure 2 is the result of the merging of two datasets. A total of 444 records were merged, by eliminating the same records. Figure 2 provides information that after several stages of
deduplication are carried out, the results will be obtained in the form of 1 single dataset as a combination of two datasets. The formed dataset contains data that has no duplication.

![Table](image)

**Fig. 2. Merging Process Results**

### 4 Conclusion

As future research, a real-time data merging process can be developed, so changes to each dataset can produce a single dataset that can be used directly. Another aspect that can be implemented is to test the performance of the results of duplication using several parameters to determine the quality of the model that has been developed.

We thank all those who have supported the publication of this article.

### References


