Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Exposé - Bachelorarbeit

Robustness of PPRL against Dataset Variation

Autor: Elias Messner
Betreuer: Florens Rohde

26.10.2022

1 Background

Record linkage is an important process for empirical studies examining personal data from several different sources, for example, medical patient data from several hospitals. Due to the lack of global IDs, the data is matched by comparing attributes like name, birth date, address, etc. Naturally, these attributes sometimes differ between the data sources, since they can change over time or contain typos. To predict if two records refer to the same real-life entity, a similarity measure is used to compare their attributes. If the similarity exceeds a threshold, the pair of records is classified as a match, otherwise a non-match.

Due to data protection laws, the data is only available with encrypted attribute values, which poses a problem in measuring the similarity between attributes. This problem is addressed by **Privacy Preserving Record Linkage** (PPRL), a procedure that uses Bloom Filter representations of the attribute values to predict a similarity. PPRL introduces a trade-off between privacy protection and linking quality and typically yields more false positives than comparing unencrypted plain text attributes.

2 Problem

Past research examines the influence of Bloom Filter size, number of hashing iterations, threshold, and other factors, on linkage quality. As of yet, there is no comprehensive research on the influence of dataset variations on linkage quality in PPRL. However, knowledge about the robustness against dataset variation is valuable, because real-world datasets may have different characteristics than those available for evaluation. This poses a problem if the linkage quality is vulnerable to dataset variations. If, on the other hand, the linkage quality is robust against dataset variations, the evaluation is more reliable, even when measured on a relatively small pool of datasets. Due to privacy protection, few datasets are available for evaluation in the field of PPRL. Given a new modification of the PPRL protocol, it is difficult to tell if a potential improvement only occurs coincidentally on a specific dataset. Therefore, this work aims to generate variations from a given dataset and evaluate the influence of these variations on the linkage quality.

The robustness of record linkage against dataset variation has already been examined in the course of a lecture project as part of the "Big-Data-Praktikum" (BDP). The students compare the linkage performance of different classifiers on various subsets. These dataset variations include subsets obtained through selection by attribute values, such as considering only specific genders, age ranges, postal codes, or names, as well as random subsets of varying sizes. They find a great influence of subset size on linkage quality and suppose there might also be additional properties of the subsets affecting the linkage quality. Building on this lecture project, this work attempts to examine the robustness of PPRL against dataset variations and to distinguish the different dataset characteristics that influence linkage quality.

3 Solution Approach

Two approaches are coming to mind, a decentralized and a centralized one. They are similar in their respective modules but different in the way these modules interact with each other. The decentralized approach works similar to a pipeline, whereas in the centralized approach the main module controls the other modules.

3.1 The Modules

Dataset Variation aka. Main Module (Java or Python)

This module is responsible for reading the whole dataset from a file or database, as well as for reading the experiment configurations from a configuration file. Its main purpose is to create the data subsets and store them in a file, making them accessible for the record linkage unit.

Since the students of the BDP project find a significant influence of subset size on linkage quality, this module focuses primarily on generating dataset variations of the same size. We can achieve this by drawing multiple same-sized random samples from an original dataset or replacing records in one subset with records from another.

Record Linkage Module (Java)

This module performs the record linkage step, which means it receives a subset and classifies all pairs as match or non-match. It allows for different similarity measures and different classifiers (threshold, supervised learning, unsupervised learning) and creates the Bloom Filters from the data received, allowing to adjust parameters such as hashing mode, hashing seed, and Bloom Filter configurations.

Record Linkage Interface (Java)

A linkage interface abstracts the record linkage module. An adapter in-between the interface and linkage module may be useful.

Linkage Evaluation Module (Java or Python)

This module gets a list of pairs classified as matches and evaluates these predictions against a list of true matches. It returns the scores for each run, namely the evaluation metrics precision, recall, F1-score, and possibly more. The true matches may be either input along with the predictions or read from a file.

Tracking Module

To keep track of the experiments and runs, as well as the scores of each run, a tracking module is needed. MLflow is a good choice since it is free and works well with Python.

3.2 Decentralized Approach

See Figure 1. The dataset variation module inputs the subsets to the record linkage module, and the latter passes the resulting pairs directly to the linkage evaluation module. The linkage evaluation module calculates the scores and stores them in MLflow.

This architecture may provide better modularity than the centralized one, although this is still debatable. For instance, both the record linkage module and the linkage evaluation module would need to know about the different experiments and runs so they can be stored in MLflow.

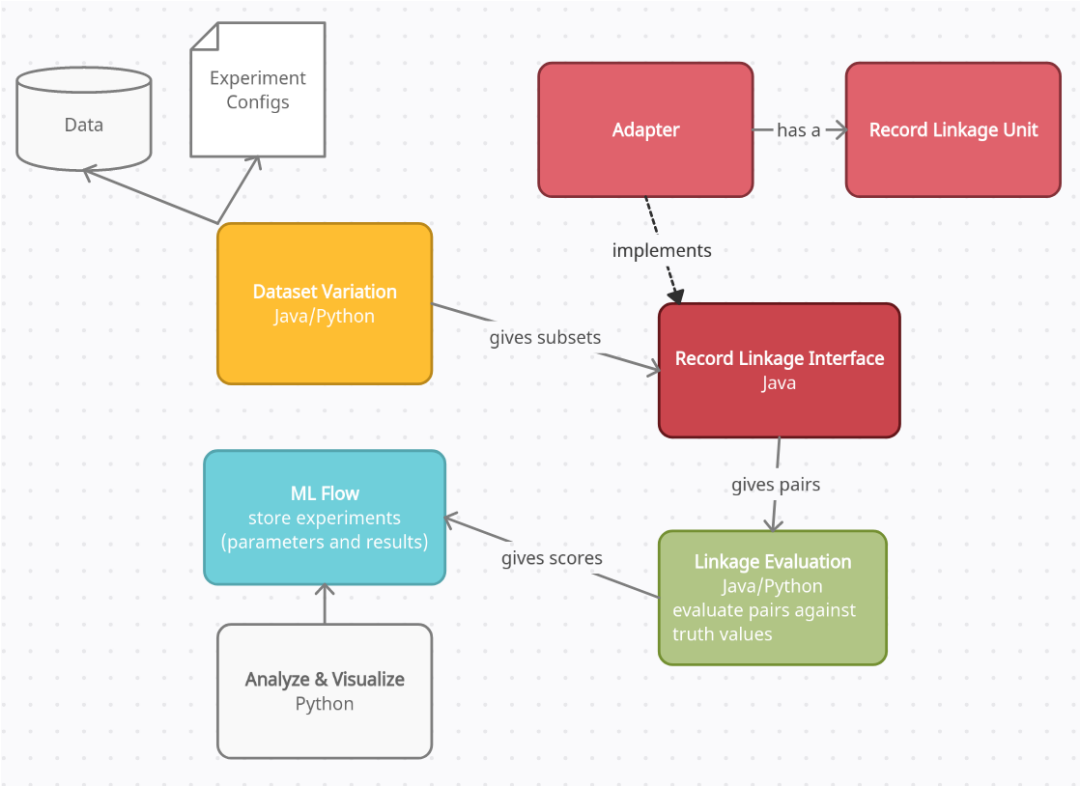

Figure 1: Decentralized Approach

3.3 Centralized Approach

See Figure 2. In this architecture, the dataset variation module functions as the main module. In addition to creating subsets, it is responsible for managing the linkage processes and the experiments and runs. Instead of passing the matched pairs on to the linkage evaluation module, the linkage module returns them to the main module. The main module then utilizes the linkage evaluation module to obtain the scores and stores them in MLflow.

This architecture is closer to the approach presented in the BDP project mentioned above. Its drawback may be implementing the God Object anti-pattern.

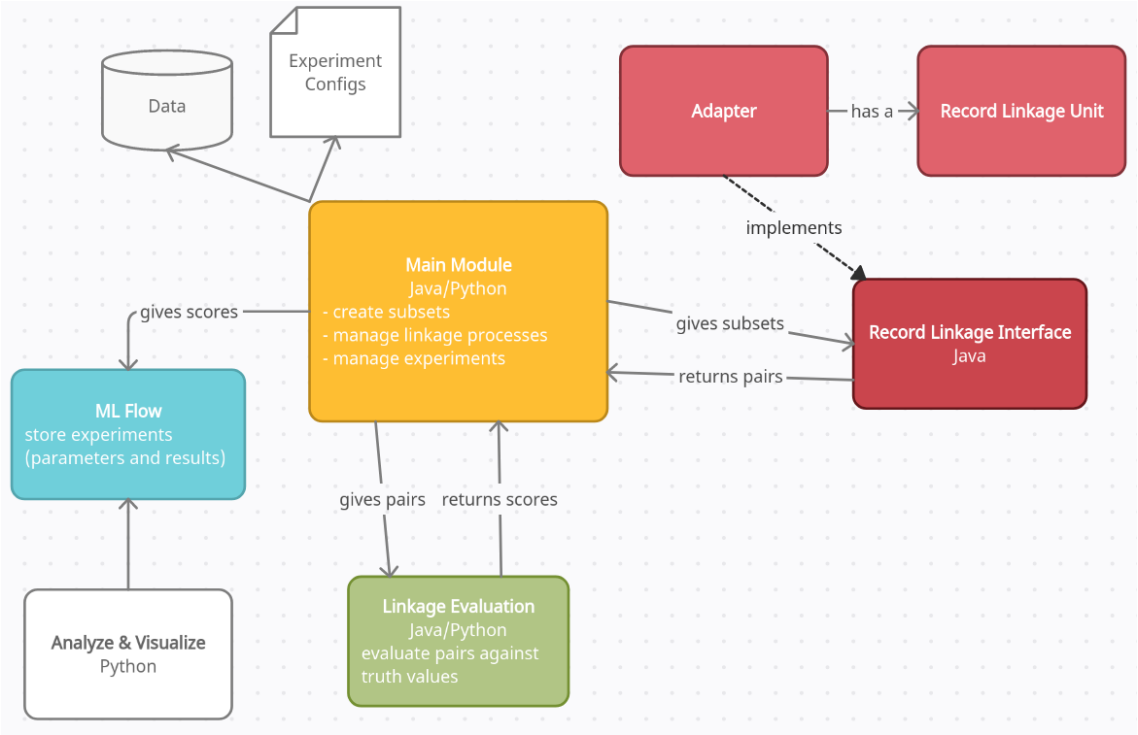

Figure 2: Centralized Approach

3.4 Further Deliberations and Open Questions

MLflow provides a well-documented Python API. Overhead may be reduced by using Python for the module that handles MLflow, namely the linkage evaluation module in the decentralized approach or the main module in the centralized approach. A combination of both approaches presented may also be worth considering.

In the BDP project, the students measure the average F1-scores for threshold values from 0.5 to 0.9. The plotted F1-score is strictly monotonically increasing, which is unexpected since usually there is an optimal threshold somewhere between 0.6 and 0.9. This phenomenon may be caused by blocking since blocking can reduce the number of false positives. However, it also increases the number of false negatives. It may be interesting for this work to make out a more realistic relationship between threshold and F1-score.