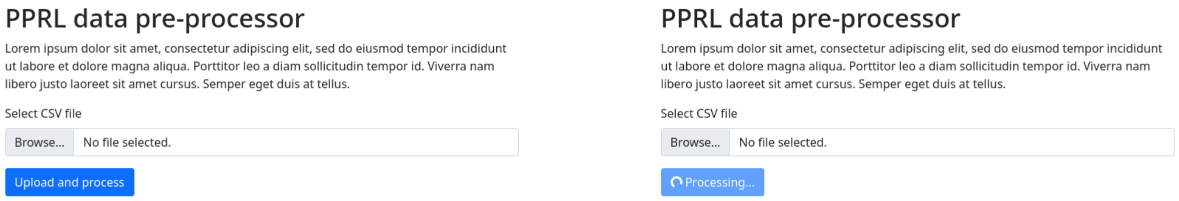
Data pre-processing for record linkage

Motivation

The main focus of our research group is to prepare and analyze medical data. One of our many fields of research concerns itself with "privacy-preserving record linkage", or PPRL for short. Medical and personal data is generated at many different locations at once. Identifying similar records between multiple datasets at various locations is a crucial step in eliminating duplicate data records, as well as merging information from different sources together. An important step in the PPRL process is data pre-processing. Records need to be standardized and normalized across all locations. For example, this ensures that names like "Strauß" and "Strauss" end up in the same form, e.g. "strauss".

Objective

Your task is to create a simple user interface, which allows a user to perform data pre-processing for PPRL from a CSV file of their choosing. An exemplary user interface is shown in Figure 1. The application that performs the pre-processing steps, called the Transformer Service, is provided to you. The focus of your work is centered around creating an interface for the user to provide a CSV file, hand off the work of pre-processing to the Transformer Service and return the results to the user in CSV format.



PPRL data pre-processor

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Porttitor leo a diam sollicitudin tempor id. Viverra nam libero justo laoreet sit amet cursus. Semper eget duis at tellus.

Select CSV file: No file selected.

(a) Idle state

PPRL data pre-processor

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Porttitor leo a diam sollicitudin tempor id. Viverra nam libero justo laoreet sit amet cursus. Semper eget duis at tellus.

Select CSV file: No file selected.

(b) Processing state

Figure 1: Example interface for uploading CSV files to pre-process

The user's CSV file is composed of a header row and a number of records, where one row represents one record. In order, these columns contain a unique identifier, first name, last name, gender and birth date. An example is shown below in Table 1.

ID	First name	Last name	Gender	Birth date
001	Max	Mustermann	Male	1987-06-05
002	Maria	Musterfrau	Female	1989-01-02
003	Claire	Grube	Female	1969-04-04

Table 1: Example structure of a CSV file

Once the user chooses the CSV file that they'd like to process, the records contained within must be sent to the Transformer Service. The Transformer Service offers a simple REST API and a single endpoint, which takes in a list of records and a list of pre-processing steps, and returns the pre-processed records as a list. While the Transformer Service supports many different pre-processing steps, only two are relevant to this application. The application must instruct the Transformer Service to apply the following steps to all attributes:

1. String normalization (removal of ligatures, reduction of whitespaces, conversion to lowercase)
2. Special character filtering (removal of special ASCII characters)

The pre-processed records returned by the Transformer Service must be converted back to CSV. The output may be offered as a file to download, or as a text field from which the user can copy and paste the pre-processed records. Applying the mentioned steps on the table shown above would yield a result similar to what's shown in Table 2.

ID	First name	Last name	Gender	Birth date
001	max	mustermann	male	19870605
002	maria	musterfrau	female	19890102
003	claire	grube	female	19690404

Table 2: Example output of the pre-processing application

Prerequisites

Using the provided CSV file

You are given an example CSV file called `pprl-transform-example-data-100.csv` with 100 randomly generated records. You may use it to test your application. The columns represent the expected input to the application. You are free to generate more CSV files with different columns, but make sure that your application is capable of processing the provided CSV file at the very least.

Running the transformer service

As mentioned before, the transformer service offers a REST API that exposes a single endpoint. Instructions on how to use this endpoint are available on Docker Hub[1].

Using Docker The recommended way to install the transformer service is to use Docker[2]. It is available as a desktop application for Windows and Mac and as an installable package for most common Linux distributions. You can start the service by running `docker run --rm -p 8080:8080 mds4ul/pprl-transform:latest` in a terminal. This will expose the API endpoint of the transformer service on port 8080 of your machine. If you wish to use a different port, e.g. 9090, change the command above like so: `-p 9090:8080`. You can stop the service by pressing `Ctrl+C` on your keyboard.

Using Java If you want to run the service without using Docker, you can run the JAR file that is provided with this assignment using Java. Install Java and make sure you're using at least version 19[3]. You can check the installed version by running `java -version` in a terminal. Start the service by changing into the directory with the executable JAR file, then running `java -jar pprl-transform-service.jar` in a terminal. This will expose the API of the transformer service on port 8080 of your machine. You can stop the service by pressing `Ctrl+C` on your keyboard.

Requirements

You may choose to use any programming language, libraries, frameworks or technologies to accomplish this task. It is possible to implement the user interface as a desktop application or as a web-based form. Refrain from using any proprietary or closed-source tools and libraries.

Use Git in your project to track changes to your files. Add a README file that describes how to install and use your project. When you're done, upload the Git repository to a service like GitHub or GitLab. Make sure that it's public and send the link to the repository to us. If you're new to Git, we recommend to use GitHub Desktop[4] to manage your Git repository. There are many resources on the internet that explain the basic commands necessary to work with Git.

[1] See `https://hub.docker.com/r/mds4ul/pprl-transform`

[2] For installation steps see `https://docs.docker.com/engine/install/`

[3] For downloadable installers see `https://adoptium.net/temurin/releases/?version=19`

[4] See `https://desktop.github.com/`