COMP3430/COMP8430 – Data Wrangling – 2021

Extra lab:: Privacy-Preserving Record Linkage Week 11

Note: There will be no interactive lab sessions for this lab. However, if you have any questions related to this lab, please post them in the Wattle forum.

Overview and Objectives

You have spent the past five labs (3 to 7) constructing the steps of a complete record linkage program. In this lab you will learn how to perform privacy-preserving record linkage (PPRL) using Bloom filters, which is covered in lecture 23. You will then conduct experiments on the data sets we were using in the earlier record linkage labs and see how linkage results differ between PPRL and non-PPRL systems, and how the different parameters and choices on Bloom filters can affect the linkage results. Make sure you view lecture 23 before going through this lab.

Preliminaries

You should start this lab by downloading from Wattle the `privacyPreservingRecordLinkage.py` Python module (available in Week 11) that can be used to perform PPRL. Once you have downloaded this Python file copy it into the folder where you keep the other record linkage Python files.

In this lab we are performing PPRL based on Bloom filters as discussed in lecture 23. For implementing Bloom filters we are going to use the `bitarray` module in Python. This module provides an object type `bitarray` which efficiently represents an array of Booleans. Bitarrays are sequence types and behave very much like usual Python lists. More information about bitarray can be found at: https://pypi.org/project/bitarray/.

Installing bitarray: Since bitarray is not included in the standard Python library, first we need to install this module. First start a Terminal. For Ubuntu users this can be done by clicking on the Main Menu (round orange-red icon on the left top menu panel), select Accessories, then Terminal. If you are using the alternative interface to Ubuntu, it can be found by clicking on Applications, then System Tools, then Terminal. A new window will pop-up with a cursor that allows text input. Type the command `pip install bitarray` followed by 'Enter' to install the bitarray package. This should install the package correctly within your user environment.

Note if you are using the cecs-student Ubuntu virtual machine the bitarray module is already installed within the anaconda distribution.

To check if the package was correctly installed, start Python in a terminal by typing `python` into the terminal window followed by 'Enter'. The prompt (character at the beginning of the line where the cursor is) should have changed to '>>>'. Then try to import the bitarray package in the Python console by typing `import bitarray` followed by 'Enter'. If you have successfully installed the bitarray package then the prompt should have changed back to '>>>', otherwise you will see an import error message. Please let us know if you encounter any problems or errors installing the bitarray package.

Lab Questions

Before you start, please have a look through the `privacyPreservingRecordLinkage.py` Python module to get an understanding of how it is structured since this is the module that runs the complete PPRL process. Similar to `recordLinkage.py`, the program `privacyPreservingRecordLinkage.py` uses two other modules we used in the record linkage program, which are `loadDataset.py` and `evaluation.py`.

Similar to `recordLinkage.py`, the whole PPRL process in the `privacyPreservingRecordLinkage.py` module is implemented in different steps. These steps include,

1. Load the data set.
2. Generate the q-gram sets for each record in the data set.
3. Generate a Bloom filter for each q-gram set.
4. Perform blocking upon Bloom filters.
5. Compare each Bloom filter pair using a similarity function.
6. Classify each Bloom filter pair as a match or a non-match.
7. Evaluate the linkage result.

Except for step 1 and 7 (which generally use functions from `loadDataset.py` and `evaluation.py`), all other steps are implemented as a separate function each in the `privacyPreservingRecordLinkage.py` module. Have a look through these functions and get a feel for how they are implemented and what the different parts are. If there is any code you do not fully understand please post a question in the Wattle forum.

Next run `privacyPreservingRecordLinkage.py` as it is. It will use some of the provided data sets, parameters, and the functions already implemented. This will show you what the output for the different steps will look like. We recommend using the small data sets with no corruptions, named `clean-A-1000.csv` and `clean-B-1000.csv`, as a way to test whether your program is working.

There are not really any implementation tasks for today's lab beyond making use of the PPRL system and experimenting with it. **Note that the focus of this lab is to understand how the different (blocking, comparison, classification) steps of the linkage process work in a privacy-preserving context, and how privacy preservation can affect the linkage quality in record linkage**.

Experiment with different attribute selections and different parameter settings in each step. Also run the linkage program on the data sets of different sizes and quality levels (clean to very dirty). Some questions you may wish to consider in your experiments include:

- Which attributes in the data set produce the best linkage results with Bloom filters? Is there a particular attribute combination which always produces good results with different parameter settings?
- What is the best choice for the q-gram length (q) in the q-gram generation step? Do different q-gram lengths produce different linkage results, or which choice produces the best results (or the best results you can find)?
- What is the best choice for the Bloom filter length and the number of hash functions in the Bloom filter generation? Does the hashing method change the linkage quality? How does the runtime change with different Bloom filter length, number of hash functions, and hashing method? Can you identify a trade-off between linkage quality and runtime based on different Bloom filter length, number of hash functions, and hashing method used?
- Which similarity function produces the best linkage results? Can you improve the linkage quality by using different similarity functions for comparing a pair of Bloom filters? What do you think are the differences between these similarity functions, and can you think of an appropriate one that is most suitable for Bloom filters?
- Do different thresholds have an effect on the number of matches? Can you improve your linkage result by changing other parameters for the same threshold?
- Are there some parameter settings or functions that are just inferior to others for all data sets and on all evaluation metrics?
- Can you spot any patterns in the results? Are there some functions that seem to work well on different data types? Do certain parameters seem to require a particular range in order to achieve reasonable results (e.g. the similarity thresholds)? Could you use these patterns to inform your choice of functions and parameter settings in the future?
- What do you think about the linkage quality that you can achieve with PPRL compared to non-PPRL? Can your PPRL system achieve linkage results that are better than the non-PPRL system? If not why?

Please note that for some parameter settings and attribute choices, the program may become very slow, especially on the larger data sets. Please terminate a run early rather than spending the whole time waiting for it to complete..