COMP3430/COMP8430 – Data Wrangling – 2019

Lab 3: Blocking for Record Linkage Week 5

Overview and Objectives

Today's lab is the first in a series of five labs during which we will gradually build a complete record linkage system. We will provide you with basic Python skeleton modules and over the next few labs you will be asked to complete the different components of the modules.

In today's lab we examine the blocking step, which is covered in lectures 13 and 14. The idea of blocking is to avoid having to compare every possible pair of records across two data sets. By using blocking we exclude from the comparison step those record pairs which are very unlikely to be matches.

Preliminaries

You should start this lab by downloading from Wattle the `comp3430_comp8430_reclink-lab-3-6.zip` archive (in Week 5) that contains the lab materials. In there you will find the skeleton modules and data sets which we will be using to complete the record linkage system. Please create a folder for the program, either on your laptop or on the lab machines, whichever you wish to use, and extract the materials there.

Before you start, please also have a look through the Python skeleton modules to get a feel for how it is structured and what the different parts are. Today we are only working with `blocking.py`, but it is good to have some understanding of how it all fits together. First look at `recordLinkage.py` since this is the module that runs the complete process.

Run `recordLinkage.py` as it is. It will use some of the provided data sets, 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. Once your program is working, apply it on the other, larger, data sets.

Lab Questions

Your tasks today are to experiment with the blocking process and to implement two blocking functions. We have already provided one blocking function, `simpleBlocking`, where records are placed into different blocks based on the value of a chosen blocking attribute (or attributes). For example, all the records with the same *Surname* will go in the same block.

We will begin the lab by reviewing the aim of blocking, and how simple blocking, Soundex, and the SLK-581 methods (see below) work. The tasks for this lab are as follows:

1. As a group, review and discuss the aim of blocking as part of the record linkage process. Discuss how simple blocking works.

   As a group, manually calulate the Soundex codes for your tutor's first and last names. Then calculate, again as a group, the SLK-581 for Brian Schmidt (our VC) and Queen Elizabeth II (their required personal details are publicly available).

   Then individually, manually calculate the Soundex code for your first name and your last name, and then calculate your SLK-581 code.

2. Next, start looking at `blocking.py` and explore how the blocking functions work (inputs, return values, etc.). Then run the blocking step on the two small data sets using both `noBlocking` and `simpleBlocking` (on attributes of your choice) and investigate how blocking affects the output. You can comment or uncomment different lines in `recordLinkage.py` to call different combinations of functions. Examine how these blocking techniques do affect the number of record pairs that are to be compared. Write down the number of blocks generated, as well as their minimum, average, and maximum sizes.

3. In the `blocking.py` module, implement the Soundex phonetic encoding, and use the Soundex value as the blocking value. In other words, all values with the same Soundex code should be placed in the same block. For a full description of Soundex, see lecture 14.

   Note: There are different specifications and implementations of Soundex, we expect you to follow the description given in lecture 14.

4. The *Statistical Linkage Key* `SLK-581` is an identifier that can be used to identify records that belong to the same person if they have the same SLK-581. As shown in the figure on the next page, SLK-581 is made up of four elements, including three letters from family name (surname or last name), two letters from given name (first name), date of birth, and gender.

   In the `blocking.py` module, implement SLK-581 as a blocking key where all records with the same SLK-581 identifier should be added to the same block. We will cover SLK-581 in more detail in lecture 16, but you can also refer to the web page at https://www.aihw.gov.au/getmedia/e1d4d462-8efa-4efa-8831-fa84d6f5d8d9/aodts-nmds-2016-17-SLK-581-guide.pdf.aspx for more details on SLK-581.

Three letters of Last name	Two letters of First name	Date of birth	Gender
XXX	XX	DDMMYYYY	N

After you have finished coding each blocking method, please repeat your experiments from task 2 on the other data sets, but using the new blocking functions. Consider both the output, i.e. the number of records per block, number of blocks, etc., and the performance information such as time taken. You may also want to print out some of the blocking keys to see what they look like, or build a frequency distribution of the blocking keys or block sizes. Think about the following questions:

- Which do you think are the best blocking functions and keys, and why?
- Can you use some of the attributes unsuitable for blocking in combination with others to improve blocking?
- Can you come up with a list of criteria for good blocking keys based on the experiments you conducted?

If time permits, there are plenty of other blocking techniques that you can look at and implement into the `blocking.py` module. Two that are commonly used in practice are canopy clustering and sorted neighbourhood blocking. If you have time please implement either of these techniques. Note that to implement sorted neighbourhood based blocking, you will need to modify the skeleton program somewhat in order to make it accept an index rather than a dictionary of blocks.