Module X: Pipeline Approaches and Deterministic ER

Rebecca C. Steorts

Agenda

- Pipeline Approach
- Deterministic Record Linkage
- Exact Matching
- Scoring Functions
- Application to XX

Load R packages

```
## Loading required package: DBI

## Loading required package: RSQLite

## Loading required package: ff

## Loading required package: bit

##
## Attaching package: 'bit'

## The following object is masked from 'package:base':
##
##     xor

## Attaching package ff

## - getOption("fftempdir")=="/var/folders/bv/xhclmwh90zg08

## - getOption("ffextension")=="ff"

## - getOption("ffdrop")==TRUE
```

Data Cleaning Pipeline

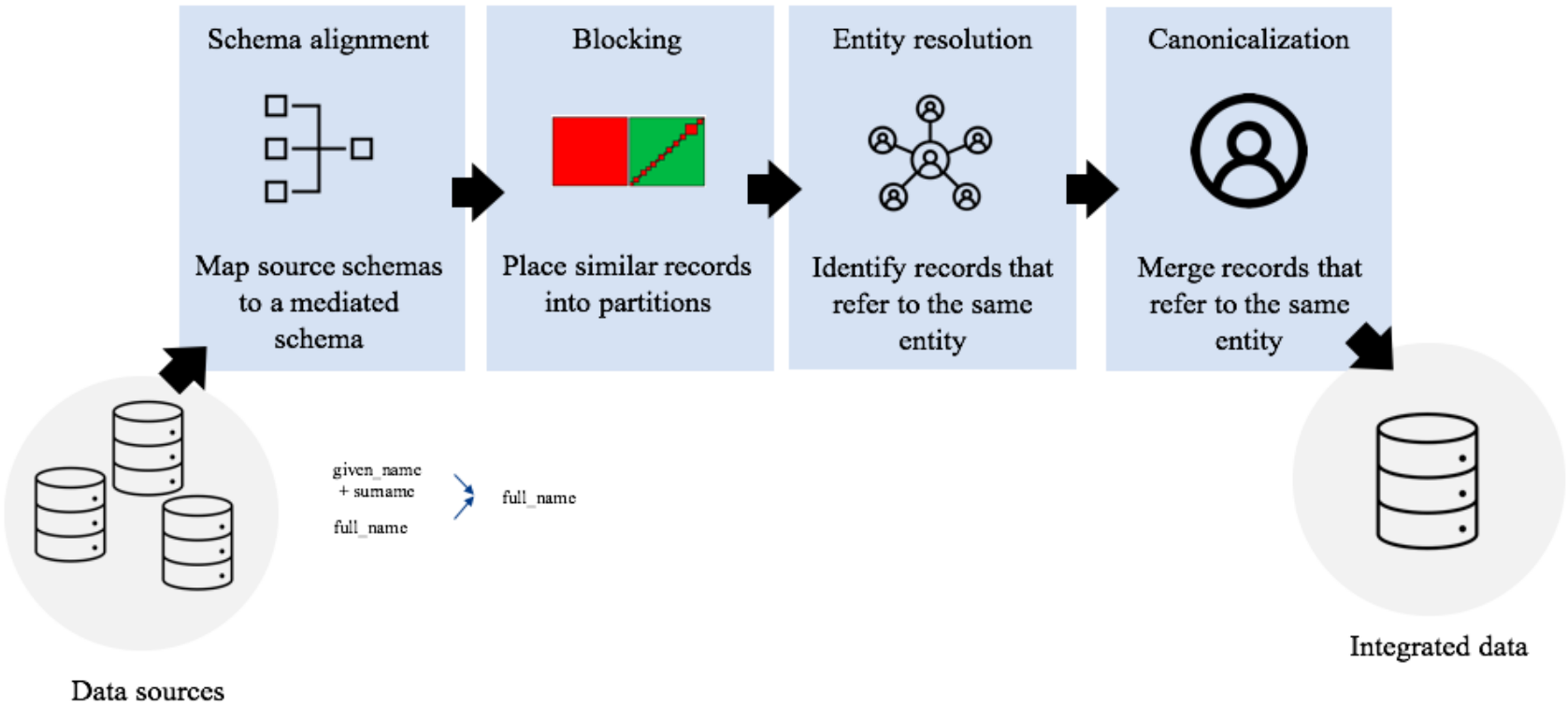

Figure 1: Data cleaning pipeline.

Deterministic Record Linkage

The most commonly used record linkage methods are based on a series of deterministic rules involving the comparison of record attributes.

Exact Matching

- Exact matching is where two record pairs are linked if they agree on all common attributes.
- An extension, off by k-matching, states that two record pairs are a match if they match on all common attributes except k, where k is an integer larger than 0.
- Exact matching (or extensions) are used when all the attributes are categorical as it tends to perform well, as opposed to when textual variables are introduced.

Scoring Rules

- Record attributes are often distorted by noise. Why would this occur?
- Linkage rules should account for such noise, distortions, and errors through scoring rules or functions.
- Examples commonly used for westernized names are the Edit (Levenshtein), Jaro, and Jaro-Winkler distance functions.

Edit (Levenshtein) distance (1966)

The Edit distance calculates the minimum number of substitutions required to transform a string s_1 into a string s_2.

Formally,

$$\text{Edit} = 1 - \frac{L}{maxLength(s_1, s_2)}.$$

Example

Consider the number of substitutions required to transform from **Adam** to **Alan.** Use the Edit distance formulate to find the similarity score that is between $[0, 1]$.

Solution

The number of substitutions required is $L = 2$.

This is normalized into a similarity function using the following:

$$\text{Edit} = 1 - \frac{L}{maxLength(s_1, s_2)} = 1 - 2/4 = 1 - 0.5 = 0.5$$

Solution

Let's verify this in R.

```
s1 <- "Adam"
s2 <- "Alan"
levenshteinSim("s1", "s2")
```

```
## [1] 0.5
```

Jaro-Winkler

- The Jaro distance (1989), called J, considered common characters and character transpositions.
- The Jaro-Winkler (1990) similarity measure, denoted JW is:

$$JW(A, B) = J(A, B) + 0.1p(1 - J(A, B))$$

where p is the # of the first four characters that agree exactly.

Example

Let's return to the example of comparing Adam and Alan.

- Here, $p = 1$.
- Given the complexity, we will calculate J and JW using `R`.

Example

```
## It seems Jaro is not supported in R
jarowinkler(s1,s2)
```

```
## [1] 0.7
```

e.g. Adam vs Alan: p=1, J= 0.67 and JW=0.7.

These work well on English names that are less than 7 characters.

Other distance functions

There are many other distance functions, such as the Jaccard, Hamming, and Cosine distances just to name a few.

Case study on El Salvador.

Let's consider a case study from El Salvador.

Talk about the data set.

Task 1

What types of string distance metrics are appropriate and which are not appropriate?

Task 2

How does exact matching perform on this data set? What about off-by-one matching?

Task 3

How would you build a decision rule for matches/non-matches based upon scoring rules. What would your scoring rule be? Write this up.

Task 4

Implement you scoring rule from Task 4 and compare it to exact and off-by-one matching. What are you findings?

Task 5

Give insights into how you might be able to improve deterministic approaches moving forward if you re-did your analysis.