Module 3: Pipeline and Deterministic ER

Rebecca C. Steorts

Reading

- Binette and Steorts (2020)
- Christen (2012), Chapters 1-3
- Edit distance: https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0

Agenda

- Pipeline Approach
- Deterministic Record Linkage
- Exact Matching
- Scoring Functions

Load R packages

```r
library(RecordLinkage)
```

Data Cleaning Pipeline

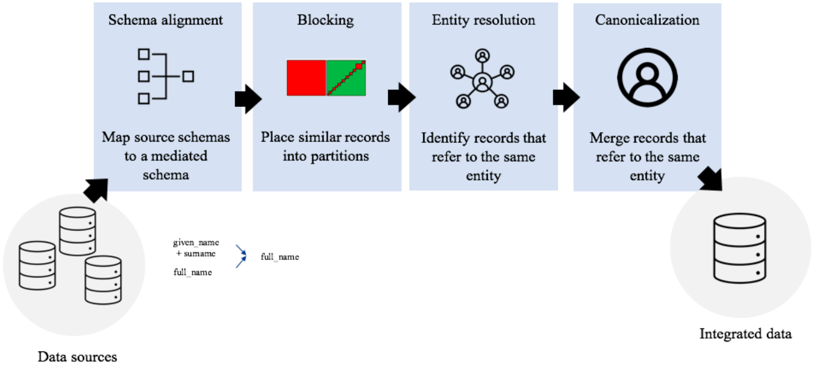

Schema alignment
Map source schemas to a mediated schema
Blocking
Place similar records into partitions
Entity resolution
Identify records that refer to the same entity
Canonicalization
Merge records that refer to the same entity
given_name
+ surname
full_name
full_name
Data sources
Integrated data

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 and off by *k* matching

- Exact matching is where two record pairs are linked if they agree on all common attributes.

Exact Matching and off by *k* matching

- Exact matching is where two record pairs are linked if they agree on all common attributes.
- Off by one matching links records pairs if they link on all common attributes (except for one).

Exact Matching and off by *k* matching

- Exact matching is where two record pairs are linked if they agree on all common attributes.
- Off by one matching links records pairs if they link on all common attributes (except for one).
- Off by two matching links record pairs if they link on all common attributes (except for two).

Exact Matching and off by k matching

- Exact matching is where two record pairs are linked if they agree on all common attributes.
- Off by one matching links records pairs if they link on all common attributes (except for one).
- Off by two matching links record pairs if they link on all common attributes (except for two).

Exact Matching and off by *k* matching

An extension, off by k-matching, states that two record pairs are a match if they match on all common attributes except at most k, where k is an integer larger than 0.

Exact Matching and off by *k* matching

An extension, off by k-matching, states that two record pairs are a match if they match on all common attributes except at most 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.

RLdata500

Consider the RLdata500 data set, removing any columns that contain missing values.

```r
library(blink) # load RLdata500
data(RLdata500)
data <- RLdata500[-c(2,4)] # Remove missing values
head(data)
```

```
##   fname_c1 lname_c1   by bm bd
## 1  CARSTEN    MEIER 1949  7 22
## 2     GERD    BAUER 1968  7 27
## 3   ROBERT HARTMANN 1930  4 30
## 4   STEFAN    WOLFF 1957  9  2
## 5     RALF  KRUEGER 1966  1 13
## 6  JUERGEN   FRANKE 1929  7  4
```

All pairs of records

Now let's consider all possible pairs of records.

```r
# create all pairs of records
pairs <- t(combn(1:nrow(RLdata500), 2))
head(pairs)
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    1    3
## [3,]    1    4
## [4,]    1    5
## [5,]    1    6
## [6,]    1    7
```

Pairwise features that disagree

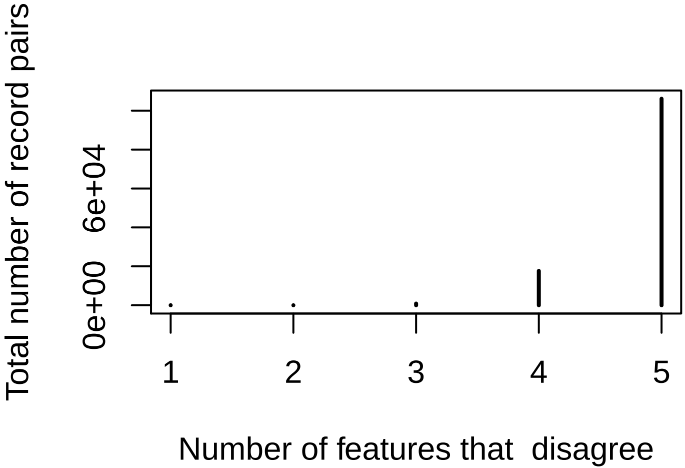
For each pair of records, compute the number of features that disagree.[1]

```r
n_disagree = sapply(1:nrow(pairs), function(i) {
  recordA = data[pairs[i,1],]
  recordB = data[pairs[i,2],]
  sum(recordA != recordB)
})
```

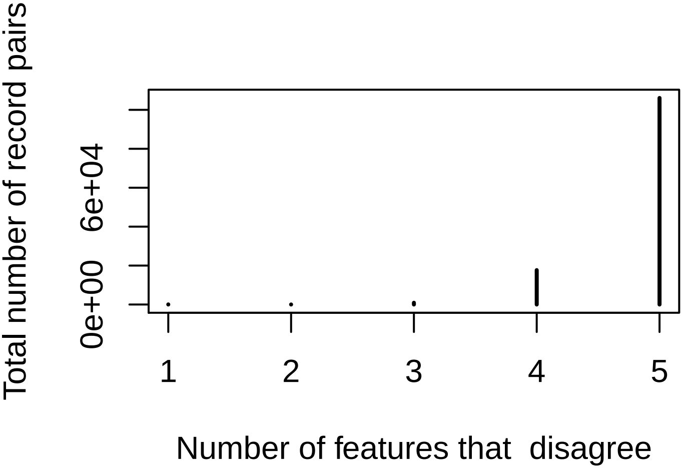
[1] This takes a few minute to compute in R. (There are more efficient ways to do this).

Distribution of Feature Disagreement Among Record Pairs

```r
plot(table(n_disagree),
     xlab="Number of features that  disagree",
     ylab="Total number of record pairs")
```



What do you observe?



Distribution of Feature Disagreement Among Record Pairs

- Observe that record pairs disagree on four or 5 features. We expect these records to not be matched.

Distribution of Feature Disagreement Among Record Pairs

- Observe that record pairs disagree on four or 5 features. We expect these records to not be matched.
- Pairs disagreeing only on 1, 2 or 3 features *might* be matches.

Distribution of Feature Disagreement Among Record Pairs

- Observe that record pairs disagree on four or 5 features. We expect these records to not be matched.
- Pairs disagreeing only on 1, 2 or 3 features *might* be matches.

Is this intuitive?

Exact Matching, Etc

Now, let's investigate exact matching (and extensions) on the `RLdata500` data set.

Exact Matching

Exact matching: Link record pairs that agree on all features.

```r
sum(n_disagree == 0)
```

```
## [1] 0
```

No pairs are exact matches!

Off by one matching

Off by 1 matching: Link record pairs that disagree only in one feature.

```
# Links
links = pairs[n_disagree <= 1, ]

# Number of estimated links
nrow(links)
```

```
## [1] 46
```

```
# Number of correctly estimated links
sum(sapply(1:nrow(links), function(i) {
  identity.RLdata500[links[i,1]] ==
    identity.RLdata500[links[i,2]]
}))
```

```
## [1] 46
```

Off by two matching

How would you extend this to off by two matching?

What do you find?

Scoring Rules

We now turn to scoring rules and how these are used in entity resolution tasks.

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.

```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

```r
## 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.

Recap

You should now be familiar with the following:

- the pipeline approach
- deterministic record linkage methods
- exact matching and extensions
- scoring functions

Discussion: How might you put these rules together to form more complex entity resolution rules?