Introduction to traditional record linkage and blocking techniques

Brenda Betancourt

Duke University
Department of Statistical Science
bb222@stat.duke.edu

February 8, 2018

Approximate matching

- Performing exact matching is very rare in practice since data is rarely noise/error free.
- Matching that allows fields to only be similar rather than exact duplicates.
- Most large-scale applications employ some level of approximate match between fields.
- Techniques for direct matching include edit distance and hashing algorithms for string data.

Example: Iterative deterministic linkage

Step 1: two records must match on SSN and one of the following:

- First and last name.
- Last name, month of birth, and sex.
- First name, month of birth, and sex.

Step 2: If SSN is missing or does not match, two records must match on last name, first name, month of birth, sex, and one of the following:

- Seven to eight digits of the SSN.
- Two or more of the following: year of birth, day of birth, middle initial, or date of death.

Approximate String Matching

Best for short strings such as person names:

- Levenshtein (edit) distance (1966): minimum number of substitutions required to transform one string into another e.g. Adam vs Alan has a distance $L = 2$, normalized as $1 - \frac{L}{maxLength} = 0.5$.

- Jaro-Winkler (1990): The Jaro distance (1989) considers common characters and character transpositions. The JW similarity measure 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 e.g. Adam vs Alan: p=1, J= 0.67 and JW=0.7.

Example: RLdata500

```r
library(RecordLinkage)
data(RLdata500)
```

```
##       fname_c1 lname_c1   by bm bd
## 314     RENATE   SCHUTE 1940 12 29
## 407     RENATE  SCHULTE 1940 12 29
## 289 CHRISTINE   PETERS 1993  2  5
## 399 CHRISTINE   PETERS 1993  2  6
## 402   CHRISTA  SCHWARZ 1965  7 13
## 462  CHRISTAH  SCHWARZ 1965  7 13
```

Example: RLdata500

```r
# Levenshtein similarity
levenshteinSim("SCHUTE", "SCHULTE")
```

```
## [1] 0.8571429
```

```r
levenshteinSim("CHRISTA", "CHRISTAH")
```

```
## [1] 0.875
```

```r
# Jaro-Winkler similarity
jarowinkler(c("SCHUTE","CHRISTA"),c("SCHULTE","CHRISTAH"))
```

```
## [1] 0.9714286 0.9750000
```

Soundex algorithm

- Generates a code that represents the phonetic pronunciation of a word, helps identifying spelling variations of names.
- The Soundex code for a name consists of a letter followed by three numerical digits:
  - the letter is the first letter of the name,
  - the digits encode the remaining consonants.
- Consonants at a similar place of articulation share the same digit
  - e.g. the labial consonants B, F, P and V are each encoded as the number 1.

Example: Soundex algorithm

```
##        fname_c1 lname_c1   by bm bd
## 314      RENATE   SCHUTE 1940 12 29
## 407      RENATE  SCHULTE 1940 12 29
## 289 CHRISTINE   PETERS 1993  2  5
## 399 CHRISTINE   PETERS 1993  2  6
## 402    CHRISTA  SCHWARZ 1965  7 13
## 462   CHRISTAH  SCHWARZ 1965  7 13
```

```r
tail(soundex(dup_set$fname_c1))
```

```
## [1] "R530" "R530" "C623" "C623" "C623" "C623"
```

```r
tail(soundex(dup_set$lname_c1))
```

```
## [1] "S300" "S430" "P362" "P362" "S620" "S620"
```

Example: Soundex algorithm

```
##      fname_c1 lname_c1   by bm bd
## 130  MICHAEL     MEYER 1988  1 31
## 147  MICHAEL      MYER 1988  1 31
## 217    HORST     MEIER 1977  6  6
## 248    HORST     MEIER 1972  6  6
## 34     HEINZ     BOEHM 1938 12 20
## 111    HEINZ    BOEHMR 1938 12 20
```

```r
head(soundex(dup_set$lname_c1))
```

```
## [1] "M600" "M600" "M600" "M600" "B500" "B560"
```

Blocking: Motivation

- Naively matching two files or finding duplicates within a file requires comparing all pairs of records.

- Infeasible for large files even when the comparisons are computationally inexpensive.

- The number of record pairs grows quadratically with the size of the dataset
  - Two files with 5,000 records $\rightarrow$ 25,000,000 comparisons!

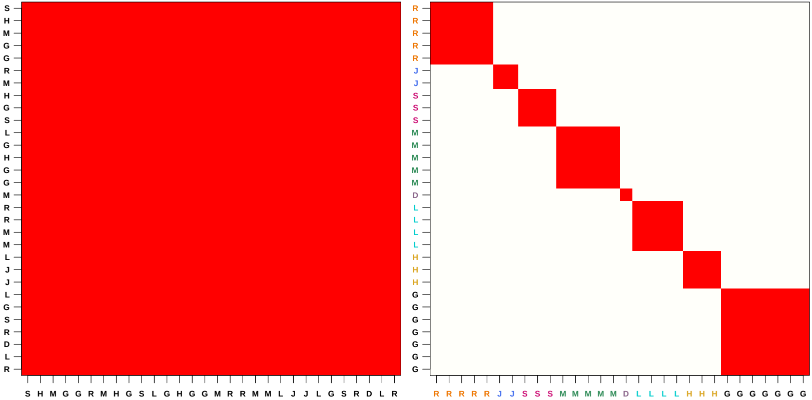
What is blocking?

Technique to reduce the comparison space:

- Filter out dissimilar record pairs that are extremely unlikely to be matches.
  - Perform record linkage only within blocks
- Traditional blocking : compare record pairs that match on one or more keys.
  - Creates a partition of the data
- Record pairs that do not meet the blocking criteria are automatically classified as non-matches.

Example: Traditional blocking

All-to-all record comparisons (left) versus partitioning records into blocks by lastname initial and comparing records only within each partition (right).



Continuation: RLdata500

```
# Record pairs for comparison
choose(500,2)
```

```
## [1] 124750
```

```
# Blocking by last name initial
last_init <- substr(RLdata500[,"lname_c1"], 1, 1)
head(last_init)
```

```
## [1] "M" "B" "H" "W" "K" "F"
```

```
# Number of blocks
length(unique(last_init))
```

```
## [1] 20
```

Continuation: RLdata500

```r
# Number of records per block
tbl <- table(last_init)
head(tbl)
```

```
## last_init
##  A  B  D  E  F  G
##  5 56  2  6 38 12
```

```r
# Block sizes can vary a lot
summary(as.numeric(tbl))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    5.75    8.00   25.00   40.00  115.00
```

Continuation: RLdata500

```r
# Number of records pairs per block
sapply(tbl, choose, k=2)
```

```
##    A    B    D    E    F    G    H    J    K    L    M
##   10 1540    1   15  703   66  496   28 1035   78 2850
##    S    T    V    W    Z
## 6555    1   21 1326   10
```

```r
# Reduction on comparison space
sum(sapply(tbl, choose, k=2))
```

```
## [1] 14805
```

Blocking caveats

- Fields can be unreliable for many applications and blocking may miss large proportions of matches i.e. increased false negatives rates.
- The frequency distribution of the values in the fields used as blocking keys will affect the size of the blocks.
- Trade-off between block sizes: true matches being missed vs computational efficiency.

How to choose the blocking key or keys

- Fields containing the fewest errors or missing values should be chosen as blocking variables e.g. clinical diagnosis in EHR.
- Understand the kinds of errors that are unlikely for a certain field or a combination of them.
- More complex blocking schemes can be constructed using conjunctions.
  - Retain only pairs which agree on either last name initial and zip code

Example: Voter Survey data

The Views of the Electorate Research (VOTER) Survey was conducted by the survey firm YouGov.

- 8,000 adults (age 18+) with internet access took the survey on-line between November 29 and December 29, 2016.
- These respondents were originally interviewed by YouGov in 2011-2012.
- Barack Obama (Democrat) won in 2012 and Donald Trump (Republican) won in 2016.

Continuation: Voter Survey data

- Demographic variables
  - Year of birth (age)
  - Gender
  - Race
  - State
  - Education level
  - Family income
- Party affiliation: democrat, republican, independent, other

Which fields are reliable for blocking in this example?

Continuation: Is race reliable?

	2012	2016
White	6244	6198
Black	654	645
Hispanic	400	397
Mixed	160	186
Other	137	167
Asian	117	118
Native American	60	59
Middle Eastern	10	12

	White	Black	Mixed	Other
White	6073	5	46	74
Black	4	627	10	10
Mixed	31	6	100	8
Other	50	4	14	62

Continuation: Is party affiliation reliable?

	Democrat	Indepen.	Republican	Not sure	Other
Democrat	2424	192	90	25	23
Indepen.	263	1929	221	16	57
Republican	39	215	1881	11	60
Not sure	48	48	54	41	5
Other	17	46	34	2	41

Blocking by disjunctions

- Produces overlapping blocks of the data.
- Using multiple keys to consider typographical or measurement errors that would exclude true matches.
  - Blocking by last name initial or zip code

A	Mary Clain	123 Oak St	90210
B	Mary Klein	123 Oak Street	90210
C	Mary Klain	123 Oak St	50210

- Reduction in false negative rates.

Example: Blocking by disjunctions

```r
# Two records must agree in either first name initial
# or bith year to be compared.
# Only 2709 pairs instead of 124750!

rpairs <- compare.dedup(RLdata500c,
blockfld = list(1, 3), #list with blocking fields
identity = identity.RLdata500)

tail(rpairs$pairs)
```

```
##       id1 id2 fname_c1 lname_c1 by bm bd is_match
## 2704 477 497        1        0  0  0  0        0
## 2705 479 483        0        0  1  1  0        0
## 2706 480 481        1        0  0  0  0        0
## 2707 480 490        1        0  0  0  0        0
## 2708 481 490        1        1  0  1  1        1
## 2709 494 497        0        0  1  1  0        0
```

Example: String comparison and blocking

```
rpairsfuzzy <- compare.dedup(RLdata500c, phonetic = FALSE,
blockfld = 3, strcmp = TRUE, strcmpfun = jarowinkler)

tail(rpairsfuzzy$pairs)
```

```
##       id1 id2  fname_c1  lname_c1 by  bm  bd is_match
## 1540 460 485 0.0000000 0.5396825  1 0.7 0.0       NA
## 1541 464 466 0.4555556 0.5396825  1 0.7 0.0       NA
## 1542 467 472 1.0000000 0.9333333  1 1.0 1.0       NA
## 1543 468 469 0.5777778 0.4666667  1 0.7 0.7       NA
## 1544 479 483 0.4370370 0.5619048  1 1.0 0.0       NA
## 1545 494 497 0.6111111 0.5026455  1 1.0 0.0       NA
```

The Fellegi-Sunter approach (1969)

- Represent every pair of records using vector of features that describe similarity between individual record fields.
    - Use string metrics (Jaro-Winkler) and edit-distances for names and strings of numbers.
- Place feature vectors for record pairs into three classes: matches (M), nonmatches (U), and possible matches.
- Let $P(\gamma|M)$ and $P(\gamma|U)$ be probabilities of observing a feature vector γ for a matched and nonmatched pair, respectively.

The Fellegi-Sunter approach (1969)

- Perform record-pair classification by calculating the ratio $w = (P(\gamma|M)/P(\gamma|U))$ for each candidate record pair.

- Establish two thresholds based on desired error levels to optimally separate the weight values for matches, possibly matches, and nonmatches.

- Drawbacks: only for two files, no trasitive closures.

Example: Fellegi-Sunter

```r
#tail(rpairs$pairs)
# Using comparison data blocking by first name initial
# and birth year
rpairs1 <- epiWeights(rpairs)

# Weights to compute thresholds for classification
head(rpairs1$Wdata)
```

```
## [1] 0.2223402 0.2223402 0.2488181 0.2488181 0.3936336 0.
```

Example: Fellegi-Sunter

```r
summary(rpairs1)
```

Weight distribution:

(0.35,0.4]	(0.4,0.45]	(0.45,0.5]	(0.55,0.6]	(0.6,0.65]
2	10	30	50	8
(0.65,0.7]	(0.7,0.75]	(0.75,0.8]	(0.8,0.85]	(0.85,0.9]
0	0	35	8	3

Example: Fellegi-Sunter

```
result <- epiClassify(rpairs1, 0.7)
summary(result)

alpha error: 0.080000 # False negative rate
beta error: 0.000000  # False positive rate
accuracy: 0.998523

Classification table:

           classification
true status    N    P    L
      FALSE 2659    0    0
      TRUE     4    0   46
```

References

- Sariyar M. and Borg A. (2010), The RecordLinkage Package: Detecting Errors in Data, The R Journal Vol. 2/2. Check big data functions of the package (>=1'000,000 record pairs).
- Steorts et al. (2014) A Comparison of Blocking Methods for Record Linkage. In: Domingo-Ferrer J. (eds) Privacy in Statistical Databases. PSD 2014. Lecture Notes in Computer Science, vol 8744. Springer, Cham.
- Dusetzina et al. Linking Data for Health Services Research: A Framework and Instructional Guide. Rockville (MD): Agency for Healthcare Research and Quality (US) (2014). An Overview of Record Linkage Methods. https://www.ncbi.nlm.nih.gov/books/NBK253312/
- Entity Resolution and Information Quality, John R. Talburt, Elsevier (2011)