

Module X: Probabilistic Blocking

Rebecca C. Steorts

Agenda

- ▶ Data Cleaning Pipeline
- ▶ Blocking
- ▶ Probabilistic Blocking
- ▶ Locality Sensitive Hashing (LSH)
- ▶ Jaccard Similarity
- ▶ Shingling
- ▶ Hash functions
- ▶ Hashed shingles
- ▶ Signatures
- ▶ Characteristic Matrix
- ▶ Minhash (Jaccard Similarity Approximation)
- ▶ Back to LSH

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("ffdrops")==TRUE
```

Data Cleaning Pipeline



Figure 1: Data cleaning pipeline.

Blocking

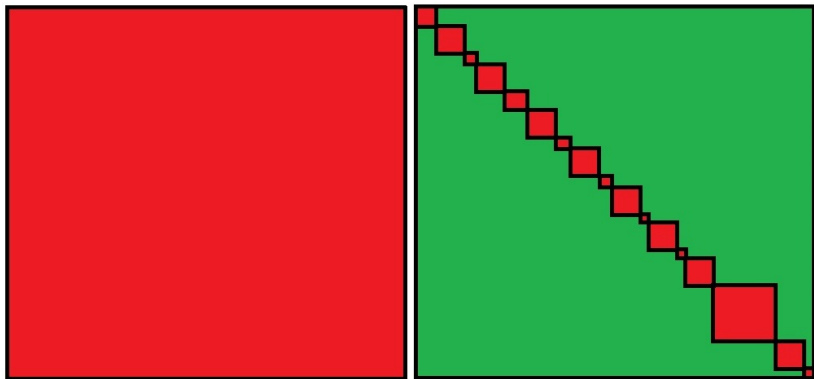


Figure 2: Left: All to all record comparison. Right: Example of resulting blocking partitions.

LSH

Locality sensitive hashing (LSH) is a fast method of blocking for record linkage that originates from the computer science literature.

Finding similar records

Our goal is to find *similar* records, where the records are assumed to be strings

How do we define *similar*?

Jaccard similarity

We will work with the *Jaccard similarity*:

$$Jac(S, T) = \frac{|S \cap T|}{|S \cup T|}.$$

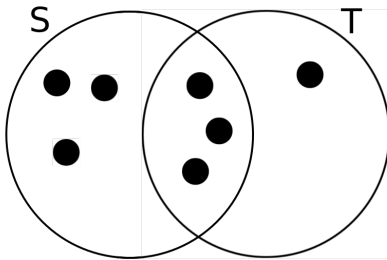


Figure 3: Two sets S and T with Jaccard similarity $3/7$. The two sets share 3 elements in common, and there are 7 elements in total.

How to represent data as sets?

We want to talk about the similarity of our data (records) \Rightarrow we need to compare sets of records!

- ▶ We can construct a set of **short strings** from the data
- ▶ This is useful because similar datasets will have many common elements (common short strings)
- ▶ We can do construct these short strings using *shingling*

k -shingling (how-to)

1. Think of the data set as a string of characters
2. A k -shingle (k -gram) is any sub-string (word) of length k found within the a record of the data set
3. Associate with each data set the set of k -shingles that appear one or more times

Let's try

Suppose our data set is the string “Hello world”, then

- ▶ the set of 2-shingles is {he, el, ll, lo, ow, wo, or, rl, ld}
- ▶ the set of 3-shingles is {hel, ell, llo, low, owo, wor, orl, rld}

Your turn

We have the following two records:

```
# load RL data  
data("RLdata500")  
  
# select only 2 records  
records <- RLdata500[129:130, c(1,3)]  
names(records) <- c("First name", "Last name")  
  
# inspect records  
kable(records)
```

	First name	Last name
129	MICHAEL	VOGEL
130	MICHAEL	MEYER

Your turn (continued)

1. Compute the 2-shingles for each record
2. Using Jaccard similarity, how similar are they?
3. What do you learn from this exercise?

Your turn solution

1. The 2-shingles for the first record are {mi, ic, ch, ha, ae, el, lv, vo, og, ge, el} and for the second are {mi, ic, ch, ha, ae, el, lm, me, ey, ye, er}
2. There are 6 items in common {mi, ic, ch, ha, ae, el} and 15 items total {mi, ic, ch, ha, ae, el, lv, vo, og, ge, lm, me, ey, ye, er}, so the Jaccard similarity is $\frac{6}{15} = \frac{2}{5} = 0.4$
3. You should have learned that this is very tedious to do by hand!

Useful packages/functions in R

(Obviously) We don't want to do this by hand most times.

Here are some useful packages in R that can help us!

```
library(textreuse) # text reuse/document similarity  
library(tokenizers) # shingles
```

```
##
```

```
## Attaching package: 'tokenizers'
```

```
## The following objects are masked from 'package:textreuse':
```

```
##
```

```
##      tokenize_ngrams, tokenize_sentences, tokenize_skip_n
```

```
##      tokenize_words
```

Shingling

We can use the following functions to create k -shingles and calculate Jaccard similarity for our data

```
# get k-shingles  
tokenize_character_shingles(x, n)  
  
# calculate jaccard similarity for two sets  
jaccard_similarity(a, b)
```


Citation Data Set

Research paper headers and citations, with information on authors, title, institutions, venue, date, page numbers and several other fields

```
library(devtools)
```

```
## Loading required package: usethis
```

```
install_github("resteorts/cora")
```

```
## Skipping install of 'cora' from a github remote, the SHA1 (70e32d5d) has not changed since last install.
```

```
## Use `force = TRUE` to force installation
```

```
library(cora)
```

```
library(ggplot2)
```

```
data(cora) # load the cora data set
```

```
str(cora) # structure of cora
```

```
## 'data.frame': 1879 obs. of 16 variables:
```

```
## $ id : int 1 2 3 4 5 6 7 8 9 10 ...
```

```
## $ title : 'noquote' chr "Inganas and M.R" NA NA NA ...
```

```
## $ book_title : 'noquote' chr NA NA NA NA ...
```

```
## $ authors : 'noquote' chr "M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O" "M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O" ...
```

```
## $ address : 'noquote' chr NA NA NA NA ...
```

```
## $ date : 'noquote' chr "1994" "1994" "1994" "1994" ...
```

```
## $ year : 'noquote' chr NA NA NA NA ...
```

```
## $ editor : 'noquote' chr NA NA NA NA ...
```

```
## $ journal : 'noquote' chr "Andersson, J Appl. Phys." "JAppl. Phys." "J Appl. Phys." "J Appl. Phys." ...
```

```
## $ volume : 'noquote' chr "76" "76" "76" "76" ...
```

```
## $ pages : 'noquote' chr "893" "893" "893" "893" ...
```

```
## $ publisher : 'noquote' chr NA NA NA NA ...
```

```
## $ institution: 'noquote' chr NA NA NA NA ...
```

```
## $ type : 'noquote' chr NA NA NA NA ...
```

```
## $ tech : 'noquote' chr NA NA NA NA ...
```

```
## $ note : 'noquote' chr NA NA NA NA ...
```

Your turn

Using the title, authors, and journal fields in the cora dataset,

1. Get the 3-shingles for each record (**hint:** use `tokenize_character_shingles`)
2. Obtain the Jaccard similarity between each pair of records (**hint:** use `jaccard_similarity`)

Your turn (solution)

```
# get only the columns we want
n <- nrow(cora) # number of records
dat <- data.frame(id = seq_len(n)) # create id column
dat <- cbind(dat, cora[, c("title", "authors", "journal")]) # get columns

# 1. paste the columns together and tokenize for each record
shingles <- apply(dat, 1, function(x) {
  # tokenize strings
  tokenize_character_shingles(paste(x[-1], collapse=" "), n = 3)[[1]]
})

# 2. Jaccard similarity between pairs
jaccard <- expand.grid(record1 = seq_len(n), # empty holder for similar
                      record2 = seq_len(n))

# don't need to compare the same things twice
jaccard <- jaccard[jaccard$record1 < jaccard$record2,]

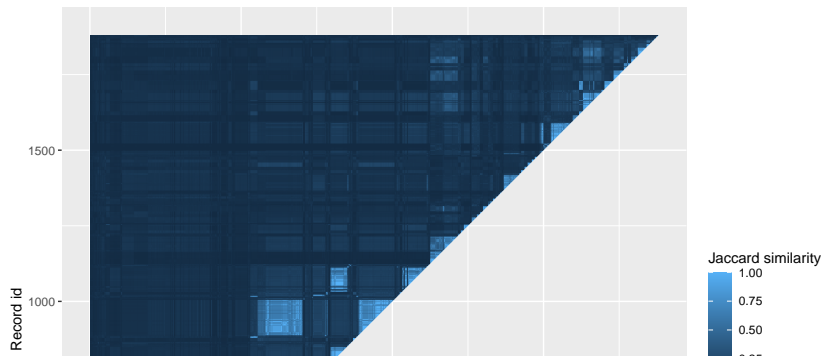
time <- Sys.time() # for timing comparison
jaccard$similarity <- apply(jaccard, 1, function(pair) {
  jaccard_similarity(shingles[[pair[1]]], shingles[[pair[2]]]) # get jaccard
})

time <- difftime(Sys.time(), time, units = "secs") # timing
```

Your turn (solution, cont'd)

plot the jaccard similarities for each pair of records

```
ggplot(jaccard) +  
  geom_raster(aes(x = record1, y = record2,  
                  fill=similarity)) +  
  theme(aspect.ratio = 1) +  
  scale_fill_gradient("Jaccard similarity") +  
  xlab("Record id") + ylab("Record id")
```



Your turn (solution, cont'd)

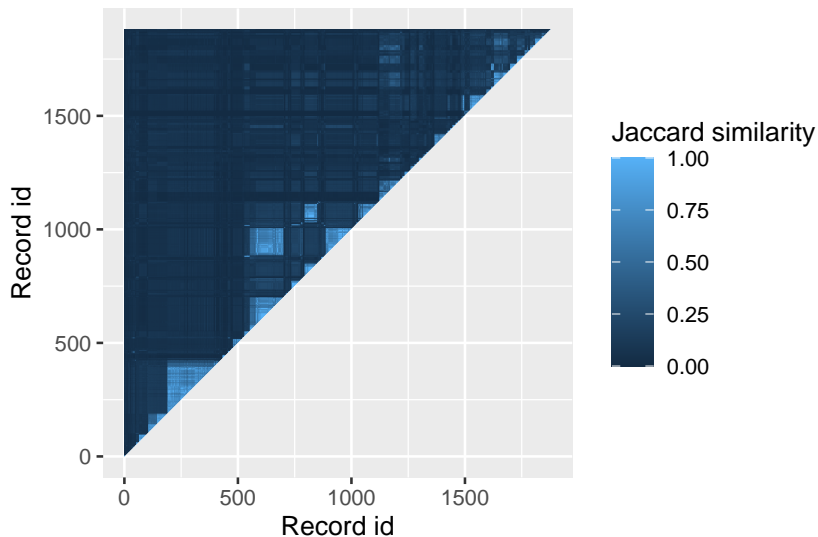


Figure 5: Jaccard similarity for each pair of records. Light blue indicates the two records are more similar and dark blue indicates less similar.

Summary

For a dataset of size n , the number of comparisons we must compute is

$$\frac{n(n-1)}{2}.$$

For our set of records, we needed to compute 1,764,381 comparisons/

For very large data sets, we need something faster (where we filter out records that are not similar).

A better approach for datasets of any realistic size is to use *hashing*, which we will look at next time.