Introduction to locality sensitive hashing

Andee Kaplan

Duke University
Department of Statistical Science
andrea.kaplan@duke.edu

February 8, 2018

Slides available at http://bit.ly/cimat-lsh

Goal and outline

Goal: Introduce locality sensitive hashing, a fast method of blocking for record linkage, and get some experience doing LSH in `R`.

1. Defining similarity
2. Representing data as sets (shingling)
3. Hashing
4. Hashing with compression (minhashing)
5. Too many pairs to compare! (LSH)
6. Evaluation

Finding similar items

- We want to find similar items
  - Maybe we are looking for near duplicate documents (plagiarism)
  - More likely, we are trying to block our data which we can later pass to a record linkage process
- How do we define *similar*?

Jaccard similarity

There are many ways to define similarity, we will use *Jaccard similarity* for this task.

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

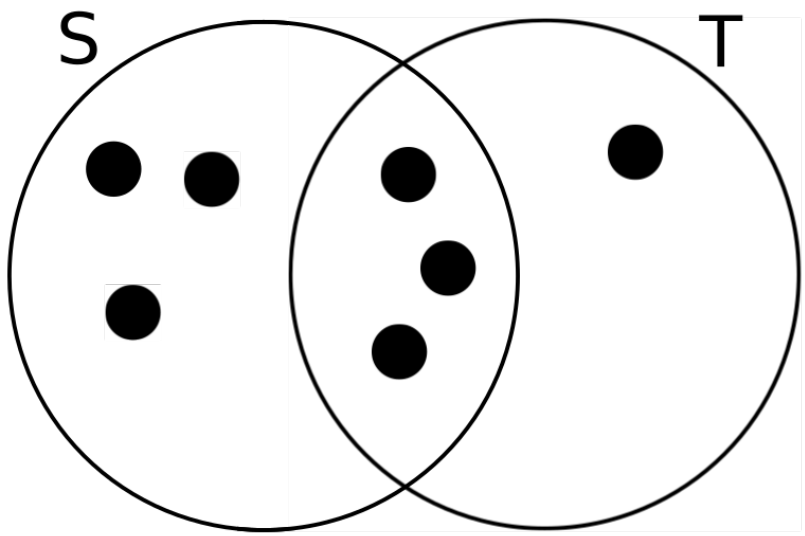


Figure: Two sets S and T with Jaccard similarity 3/7. The two sets share 3 elements in common, and there are 7 elements in total.

How to represent data as sets

We want to talk about similarity of data $\Rightarrow$ we need sets to compare!

- One way is to construct from the data the set of **short strings** that appear within it
- Similar documents/datasets will have many common elements, i.e. many commong short strings
- We can do construct these short strings using *shingling*

k-shingling (how-to)

1. Think of a document or record as a string of characters

2. A k-shingle (k-gram) is any sub-string (word) of length k found within the document or record

3. Associate with each document or record the set of k-shingles that appear one or more times within it

Let's try

Suppose our document is the string "Hello world" and $k = 2$, 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:

	First name	Last name
129	MICHAEL	VOGEL
130	MICHAEL	MEYER

1. Compute the 2-shingles for each record

2. Using Jaccard similarity, how similar are they?

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 16 items total
   $\{$mi, ic, ch, ha, ae, el, lv, vo, og, ge, el, lm, me, ey, ye, er$\}$, so the Jaccard similarity is $\frac{6}{16} = \frac{3}{8} = 0.375$

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!

```
# detecting text reuse and document similarity + shingles
library(textreuse)
library(tokenizers)
```

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)
```

Example data

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

```
library(RLdata)
data(cora)
str(cora)
```

```
## 'data.frame':    1879 obs. of  16 variables:
##  $ id          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ title       :Class 'noquote'  chr [1:1879] "Inganas and M.R" NA NA NA ...
##  $ book_title :Class 'noquote'  chr [1:1879] NA NA NA NA ...
##  $ authors     :Class 'noquote'  chr [1:1879] "M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlr
##  $ address     :Class 'noquote'  chr [1:1879] NA NA NA NA ...
##  $ date        :Class 'noquote'  chr [1:1879] "1994" "1994" "1994" "1994" ...
##  $ year        :Class 'noquote'  chr [1:1879] NA NA NA NA ...
##  $ editor      :Class 'noquote'  chr [1:1879] NA NA NA NA ...
##  $ journal     :Class 'noquote'  chr [1:1879] "Andersson, J Appl. Phys." "JAppl. Phys." "J Appl. Phys."
##  $ volume      :Class 'noquote'  chr [1:1879] "76" "76" "76" "76" ...
##  $ pages       :Class 'noquote'  chr [1:1879] "893" "893" "893" "893" ...
##  $ publisher   :Class 'noquote'  chr [1:1879] NA NA NA NA ...
##  $ institution:Class 'noquote'  chr [1:1879] NA NA NA NA ...
##  $ type        :Class 'noquote'  chr [1:1879] NA NA NA NA ...
##  $ tech        :Class 'noquote'  chr [1:1879] NA NA NA NA ...
##  $ note        :Class 'noquote'  chr [1:1879] 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`).

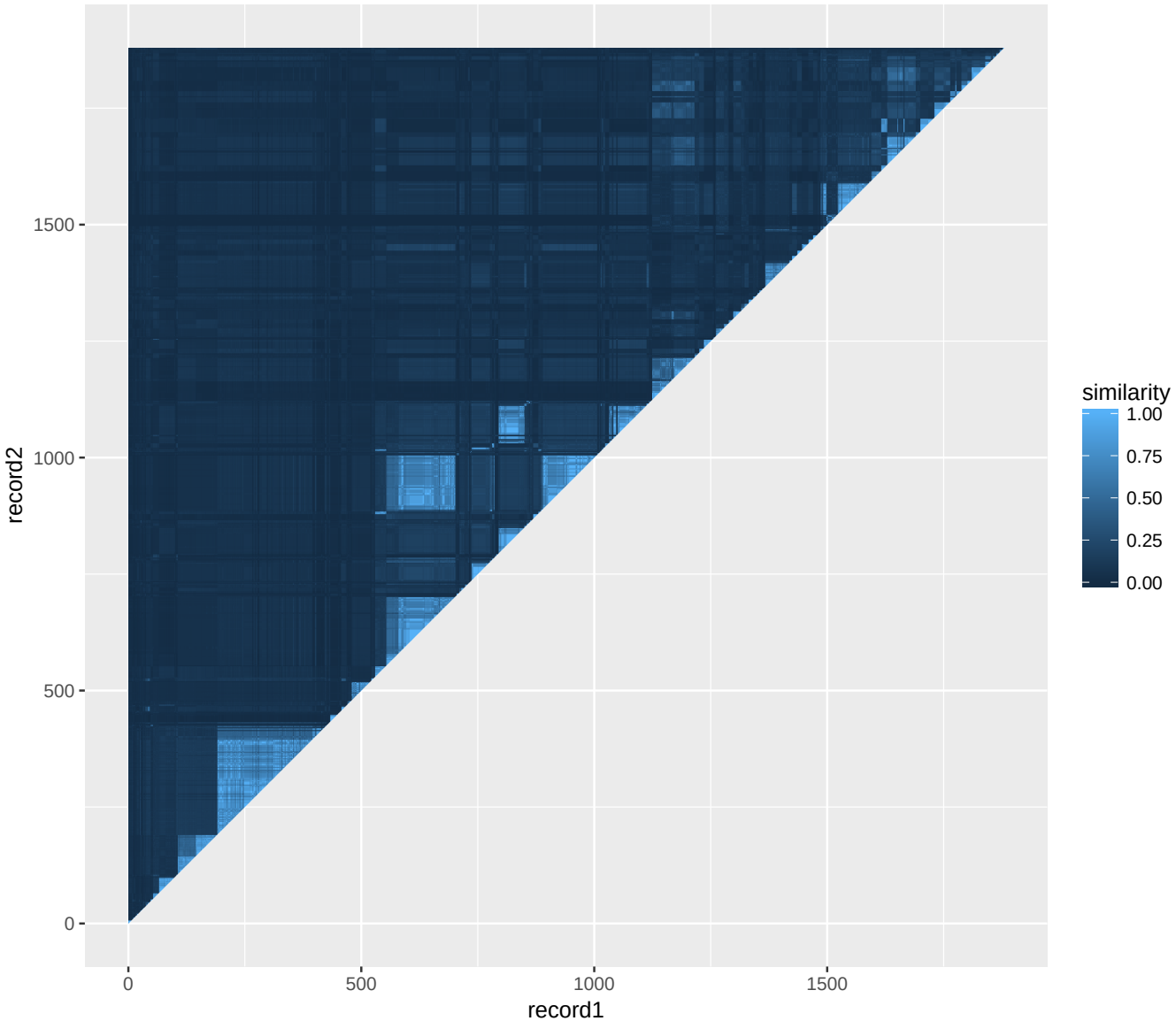
2. Obtain the Jaccard similarity between each pair of records (**hint:** use `jaccard_similarity`).

Your turn solution

```
# get only the columns we want
dat <- cora[, c("title", "authors", "journal")]

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

# 2. Jaccard similarity between pairs
jaccard <- expand.grid(record1 = seq_len(nrow(dat)),
                       record2 = seq_len(nrow(dat)))

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

time <- Sys.time()
jaccard$similarity <- apply(jaccard, 1, function(pair) {
  jaccard_similarity(shingles[[pair[1]]], shingles[[pair[2]]])
})
time <- difftime(Sys.time(), time, units = "secs")
```

This took took 150.11 seconds $\approx$ 2.5 minutes

Your turn solution (cont'd)



Hashing

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
- A better approach for datasets of any realistic size is to use *hashing*

Hash functions

- Traditionally, a *hash function* maps objects to integers such that similar objects are far apart
- Instead, we want special hash functions that do the **opposite** of this, i.e. similar objects are placed closed together!

Definition: Hash function

Hash functions $h()$ are defined such that

> *If records A and B have high similarity, then the probability that $h(A) = h(B)$ is **high** and if records A and B have low similarity, then the probability that $h(A) \neq h(B)$ is **high**.*

Hashing shingles

Instead of storing the strings (shingles), we can just store the *hashed values*

These are integers, they will take less space

```
# instead store hash values (less memory)
hashed_shingles <- apply(dat, 1, function(x) {
  string <- paste(x, collapse=" ")
  shingles <- tokenize_character_shingles(string, n = 3)[[1]]
  hash_string(shingles)
})
```

This took up 6.38256×10^5 bytes, while storing the shingles took 7.36544×10^6 bytes. However, the whole pairwise comparison still took the same amount of time (≈ 2.58 minutes).

Similarity preserving summaries of sets

Characteristic matrix

Minhashing

LSH (avoid pairwise comparisons)

Banding and buckets

Your turn

banding in R

Putting it all together

Choosing shingle size - somewhere?

“Easy” LSH in R

Evaluation

Your turn

perform LSH and evaluate how we did