Homework 3 Solutions: Locality Sensitive Hashing

STA 325

2024-09-12

This code may fix the true unique id.

```
n <- nrow(cora)

# initialize each row to be its own "group"
unique_ids <- 1:n

# update "groups" as iterate over rows
# know that assigning the new ID will be a valid group since id1 < id2 for all rows
# the unique_id for indexes corresponding to cora_ids will be the same if they
#   are really the same entity
for (row in 1:nrow(cora_gold)) {
  id1 <- cora_gold[row,1]
  id2 <- cora_gold[row,2]
  unique_ids[id2] <- unique_ids[id1]
}

# count number of unique entities
n_unique <- length(unique(unique_ids))

# map the raw group IDs to a simpler group IDs format
# allows the unique_id to only go from 1 to n_unique instead of 1 to n
map <- setNames(1:n_unique, unique(unique_ids))

# apply the mapping
unique_ids <- map[as.character(unique_ids)]

# create dataframe storing unique IDs
unique_df <- data.frame(
  cora_id = 1:n,
  unique_id = unique_ids
)
head(unique_df)
```

```
##   cora_id unique_id
## 1       1         1
## 2       2         1
## 3       3         1
## 4       4         1
## 5       5         1
## 6       6         1
```

```
# VERIFY: all matches have the same unique ID
# the result of this should be 0
```

```
cora_gold %>%
  left_join(unique_df, by = c("id1" = "cora_id")) %>%
  left_join(unique_df, by = c("id2" = "cora_id"), suffix = c(".1",".2")) %>%
  filter(unique_id.1 != unique_id.2) %>%
  nrow()
```

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

```
# VERIFY: the number of matches in cora_gold equals the number of matches we
#   would expect from these unique IDs
# the result of this should be TRUE
sum(choose(table(unique_df$unique_id),2)) == nrow(cora_gold)
```

```
## [1] TRUE
```

Consider the cora citation data set and load the data set with an column id as we did in class. Code is provided below.

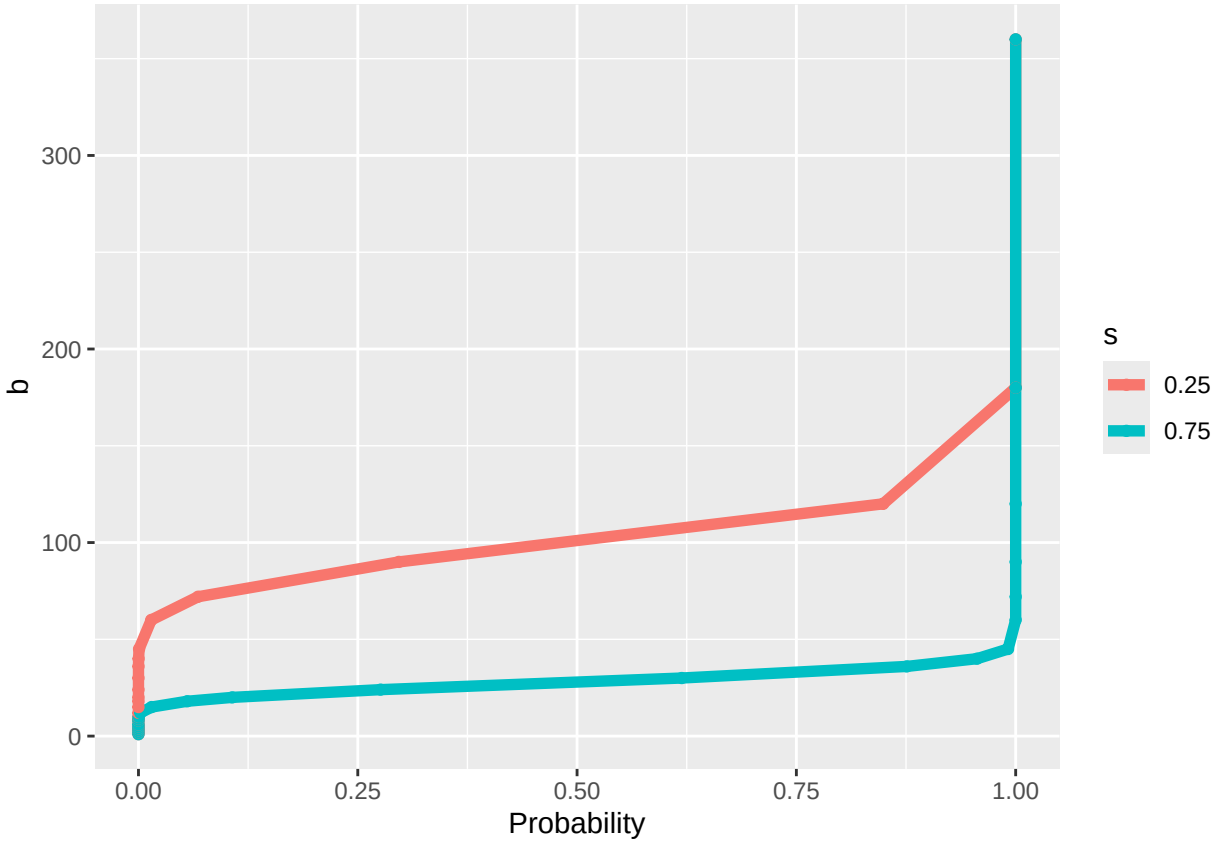
```
# get only the columns we want
# number of records
n <- nrow(cora)
# create id column
dat <- data.frame(id = seq_len(n))
# get columns we want
dat <- cbind(dat, cora[, c("title", "authors", "journal")])
```

Perform the LSH approximation as we did in class using the `textreuse` package via the functions `minhash_generator` and `lsh` (so we don't have to perform it by hand). Again, this code is provided for you given that it was done in class to make it a bit easier. Feel free to play around with this on your own. We will assume that m = 360, b = 90, and the number of shingles is 3 for this assignment.

Find the number of buckets or bands to use

```
library(numbers)
m <- 360
b <- 90
#m <- 600
#b <- 20
bin_probs <- expand.grid(s = c(.25, .75), h = m, b = divisors(m))
#bin_probs
# choose appropriate num of bands and number of random permutations m (tuning parameters)
bin_probs$prob <- apply(bin_probs, 1, function(x) lsh_probability(x[["h"]], x[["b"]], x[["s"]]))
# plot as curves
ggplot(bin_probs) +
  geom_line(aes(x = prob, y = b, colour = factor(s), group = factor(s)), linewidth = 2) +
  geom_point(aes(x = prob, y = b, colour = factor(s)), linewidth = 3) +
  xlab("Probability") +
  scale_color_discrete("s")
```

```
## Warning in geom_point(aes(x = prob, y = b, colour = factor(s)), linewidth = 3):
## Ignoring unknown parameters: `linewidth`
```




```
# create the minhash function
minhash <- minhash_generator(n = m, seed = 02082018)
```

Build corpus and perform shingling

```
head(dat)
```

```
##   id           title
## 1  1 Inganas and M.R
## 2  2            <NA>
## 3  3            <NA>
## 4  4            <NA>
## 5  5            <NA>
## 6  6            <NA>
##                                                                                 authors
## 1                                M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O
## 2 M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O. Inganas and M.R.   Andersson
## 3 M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O.  Inganas and M.R.  Andersson
## 4   M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O. Inganas and M.R. Andersson
## 5  M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O.  Inganas and M.R. Andersson
## 6   M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O. Inganas and M.R. Andersson
##                    journal
## 1 Andersson, J Appl. Phys.
## 2             JAppl. Phys.
## 3            J Appl. Phys.
## 4             J Appl.Phys.
```

```
## 5            J Appl. Phys.
## 6             J Appl.Phys.
```

```
# build the corpus using textreuse
docs <- apply(dat, 1, function(x) paste(x[-1], collapse = " ")) # get strings
names(docs) <- dat$id # add id as names in vector
corpus <- TextReuseCorpus(text = docs, # dataset
                          tokenizer = tokenize_character_shingles, n = 3,
                          simplify = TRUE, # shingles
                          progress = FALSE, # quietly
                          keep_tokens = TRUE, # store shingles
                          minhash_func = minhash) # use minhash
head(minhashes(corpus[[1]]))
```

```
## [1] -2064207635 -2111669997 -2125164234 -2097867716 -2144942958 -2116065215
```

```
length(minhashes(corpus[[1]]))
```

```
## [1] 360
```

Note that all our records are now represented by 360 randomly selected and hashed shingles. Comparing these shingles are equivalent to finding the Jaccard similarity of all the record pairs. We still have an issue of all the pairwise comparison.

Find buckets, candidate records, and Jaccard similarity

Now, we find the buckets, candidates records, and calculate the Jaccard similarity for the candidate records (in the buckets)

```
# perform lsh to get buckets
buckets <- lsh(corpus, bands = b, progress = FALSE)
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## i Please use `gather()` instead.
## i The deprecated feature was likely used in the textreuse package.
##   Please report the issue at <https://github.com/ropensci/textreuse/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# grab candidate pairs
candidates <- lsh_candidates(buckets)
# get Jaccard similarities only for candidates
lsh_jaccard <- lsh_compare(candidates, corpus,
                           jaccard_similarity, progress = FALSE)
head(buckets)
```

```
## # A tibble: 6 x 2
##   doc   buckets
##   <chr> <chr>
## 1 1     fb93d6f4c56666ec8210570af8e8edd0
## 2 1     cf942dbe840d4365cf182ea24d6951c8
## 3 1     e9535be0f24e39103ba1f11442cc170e
## 4 1     52a293069e3920a0f56e38a0f0c6af37
## 5 1     b751d8b2d24bec53a78b3043dc017837
## 6 1     0ce07d00e8c2e811204352b51229ed18
```

```
dim(buckets)
```

```
## [1] 169110      2
```

```
length(unique(buckets))
```

```
## [1] 2
```

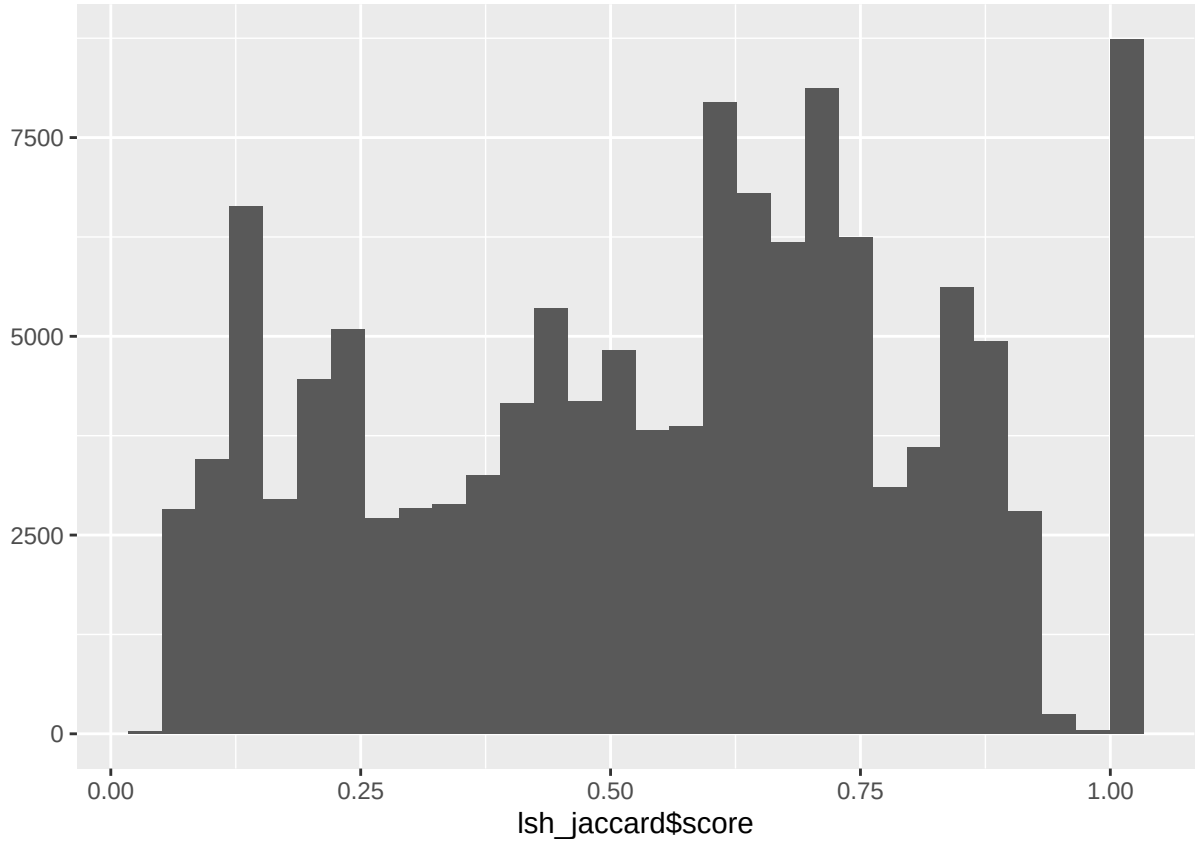
```
head(lsh_jaccard)
```

```
## # A tibble: 6 x 3
##   a     b     score
##   <chr> <chr> <dbl>
## 1 1     2     0.865
## 2 1     3     0.865
## 3 1     4     0.865
## 4 1     5     0.865
## 5 1     6     0.865
## 6 1     7     0.865
```

We now plot the Jaccard similarities that are candidate pairs (under LSH)

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



1. Calculate the reduction ratio from the total number of record comparisons (N choose 2) compared to those under locality sensitive hashing (above).

```
# calculate reduction ratio
# n is the number of total records in the data set
# blocks are the candidate record pairs that have been hashed
# to the same bucket
reduction_ratio <- function(n, blocks) {
return(1 - nrow(blocks) / choose(n, 2))
}

print(rr <- reduction_ratio(n, candidates))
```

```
## [1] 0.9276171
```

2. Find the pairwise precision and recall under locality sensitive hashing. There are two places where we have ground truth. Note that cora_gold contains record pairs that are true matches; cora_gold_update contains a unique identifer alternatively. You will need to write your own code for this.

One way to find the pairwise precision and recall is as follows:

```
# vector of candidate lsh pairs
lsh_pairs <- paste(lsh_jaccard$a, lsh_jaccard$b)
# vector of ground truth record pairs
cora_gold_pairs <- paste(cora_gold$id1, cora_gold$id2)

true_positive <- sum(lsh_pairs %in% cora_gold_pairs) #  58629

# lsh predicts a record pair but it's not a true pair
false_positive <- sum(!(lsh_pairs %in% cora_gold_pairs)) # 69082

# lsh predicted these to be not matches but they are true matches
false_negative <- sum(!(cora_gold_pairs %in% lsh_pairs)) # 5949

recall <- true_positive / (true_positive + false_negative)
precision <- true_positive / (true_positive + false_positive)

recall
```

```
## [1] 0.9078788
```

```
precision
```

```
## [1] 0.4590756
```