Homework 3: Locality Sensitive Hashing

STA 325

2024-09-12

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

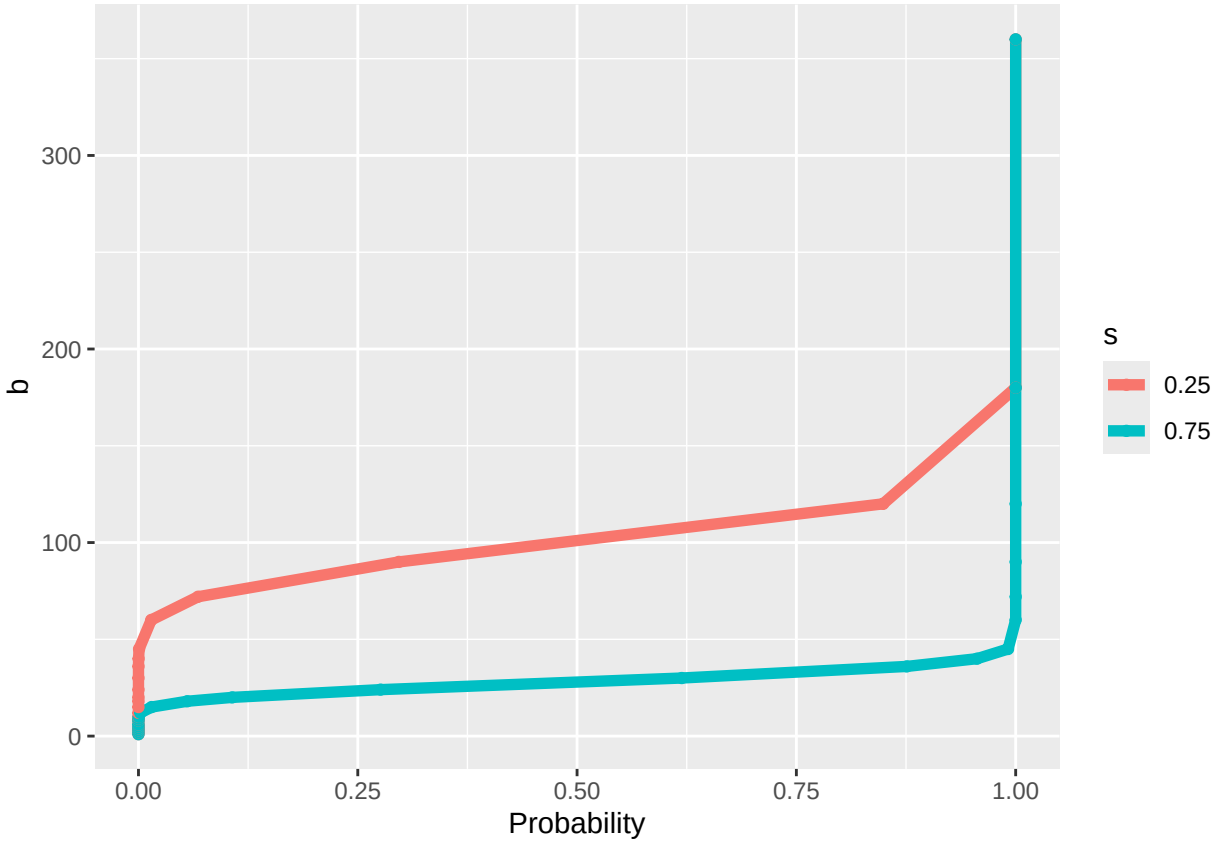
```
# get only the columns we want
# number of records
n <- nrow(cora)
# create id column
dat <- data.frame(id = seq_len(n))
# get columns we want
dat <- cbind(dat, cora[, c("title", "authors", "journal")])
```

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
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)
b <- 90
```

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] -2129559086 -2105149779 -2057649376 -2075639297 -2117081502 -2076751502
```

```
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     accb8959a23f42572d622bf3ba561176
## 2 1     5da7cfceeb2b151788611bed37096c7b
## 3 1     e10b6d7fdd9cc35fe7aeb994dfe43714
## 4 1     2a5b182348b44157eb4d0325655cee55
## 5 1     2da285eeafbe173da21f9a70bb895542
## 6 1     84e1aa3e4afc721d4e7b28c0bcb165a4
```

```
dim(buckets)
```

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

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

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

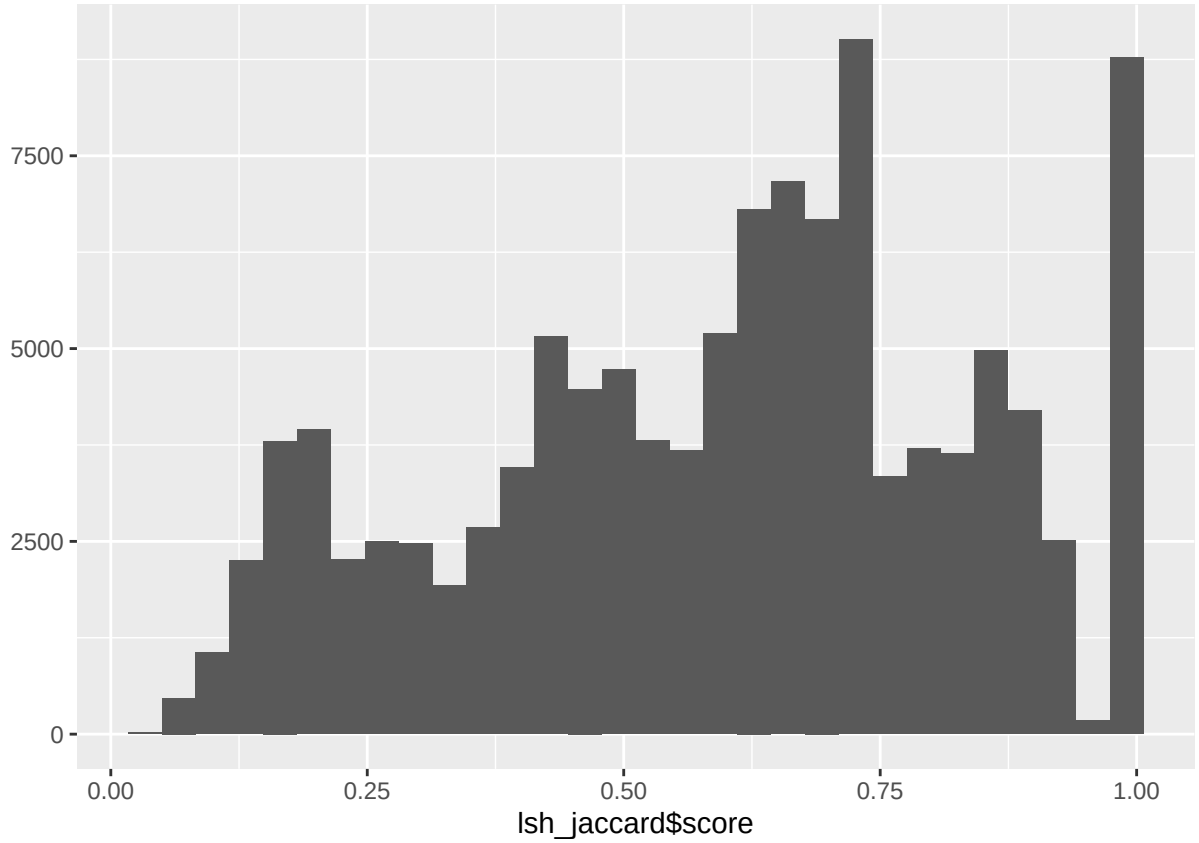
```
head(lsh_jaccard)
```

```
## # 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
(choose(1879, 2) - nrow(candidates)) / choose(1879, 2)
```

```
## [1] 0.9348984
```

The reduction ratio is 0.935.

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.

```
# Concatenate the pairs together
cora_gold$pairs <- paste(cora_gold$id1, cora_gold$id2)
lsh_jaccard$pairs <- paste(lsh_jaccard$a, lsh_jaccard$b)

# Calculate true positives
TP <- length(intersect(cora_gold$pairs, lsh_jaccard$pairs))

# Calculate false positives
FP <- length(setdiff(lsh_jaccard$pairs, cora_gold$pairs))
```

```r
# Calculate false negatives
FN <- length(setdiff(cora_gold$pairs, lsh_jaccard$pairs))

# Calculate precision
TP / (TP + FP)
```

```
## [1] 0.5108563
```

```r
# Calculate recall
TP / (TP + FN)
```

```
## [1] 0.9086531
```

Precision is 0.511 while recall is 0.909.

3. We can further reduce the problem by filtering out candidate pairs of records below a threshold t that are unlikely to be matches. For example, assume $t = 0.8$. Filter out all record pairs below the threshold of 0.8. We will call this locality sensitive hashing with filtering/thresholding.

```r
# Filter out record pairs below 0.8 threshold
lsh_jaccard <- filter(lsh_jaccard, score > 0.8)
```

4. Under lsh with t = 0.8, re-compute the precision, recall, and reduction ratio.

```r
# Calculate true positives
TP <- length(intersect(cora_gold$pairs, lsh_jaccard$pairs))

# Calculate false positives
FP <- length(setdiff(lsh_jaccard$pairs, cora_gold$pairs))

# Calculate false negatives
FN <- length(setdiff(cora_gold$pairs, lsh_jaccard$pairs))

# Calculate precision
TP / (TP + FP)
```

```
## [1] 0.8030143
```

```r
# Calculate recall
TP / (TP + FN)
```

```
## [1] 0.3184676
```

```r
# Calculate reduction ratio
(choose(1879, 2) - nrow(lsh_jaccard)) / choose(1879, 2)
```

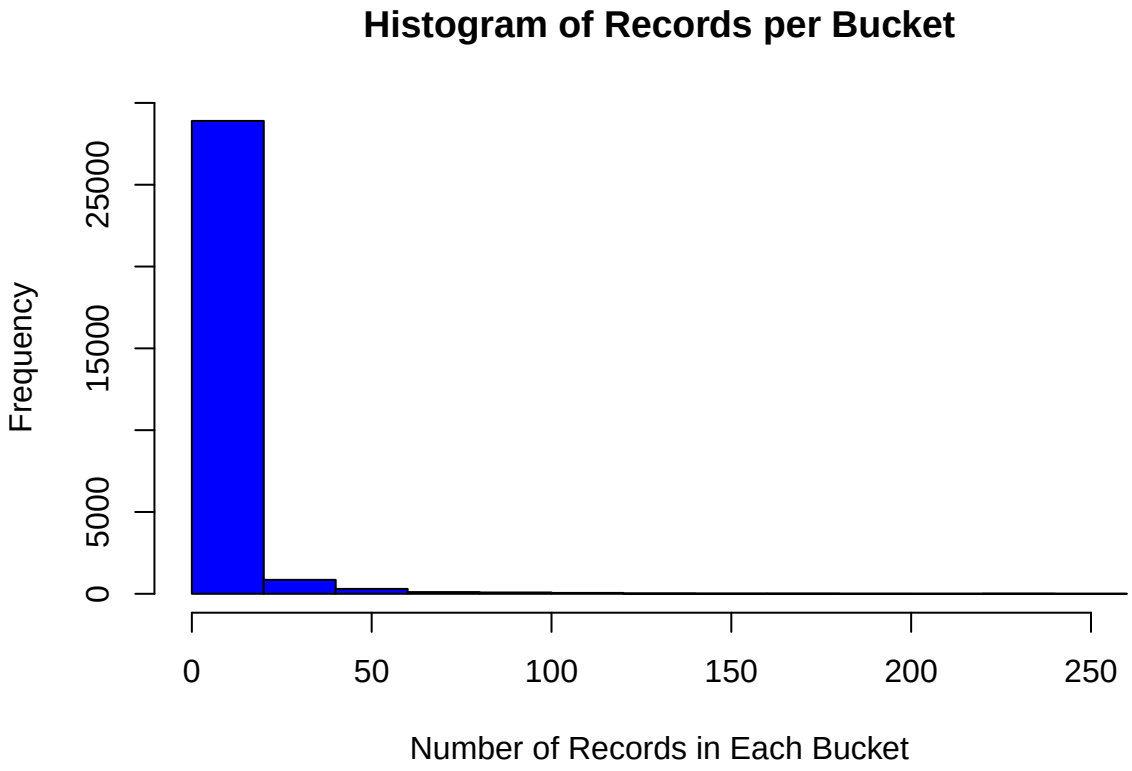
```
## [1] 0.9854844
```

Precision is 0.803 while recall is 0.318. The reduction ratio is 0.985.

5.

i. Describe what the blocks look like from this method? Hint: Try looking at a histogram of the number of records in each bucket.

```
# Create a histogram of # records in each bucket
buc_counts <- table(buckets$buckets)
hist(as.numeric(buc_counts),
     breaks = 10,
     main = "Histogram of Records per Bucket",
     xlab = "Number of Records in Each Bucket",
     col = "blue")
```



The blocks range from containing 1 record all the way to 250+ records. The total number of records in all of the blocks far exceeds the number of records in cora (almost 100x larger). A majority of blocks contain less than 10 records each.

ii. Are the blocks non-overlapping or overlapping? Hint: Part (i) should help you determine an answer to this. overlap

The blocks are overlapping because a single record can be placed in multiple blocks.

iii. Describe some advantages and disadvantages of the LSH method that you see from using it practically.

Advantages: The LSH method significantly reduces the number of pairwise comparisons which is crucial for very large datasets. Additionally, this method takes up less memory because only the hashed values need to be stored.

Disadvantages: The LSH method introduces some trade offs when you account for precision and recall before and after filtering/thresholding. The method is better when recall is the priority before filtering. The method is better when precision is the priority after filtering. This is a disadvantage because there are times you might want both precision and recall to be high. Additionally, tuning parameters such as number of bands and hash function can be tricky and optimal setting can vary from dataset to dataset.