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.

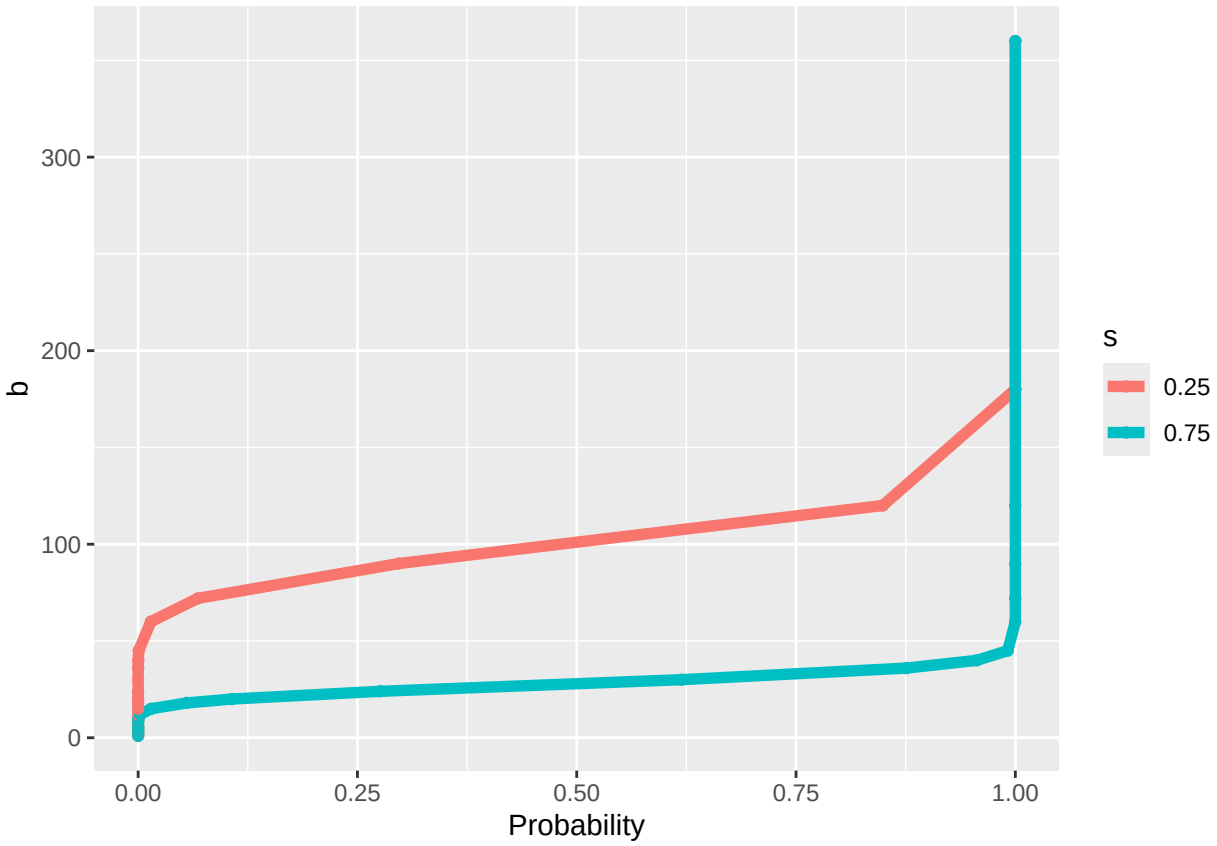
```
# get only the columns we want
# number of records
n <- nrow(cora)
# create id column
dat <- data.frame(id = seq_len(n))
# get columns we want
dat <- cbind(dat, cora[, c("title", "authors", "journal")])
```

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)

# 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))
```

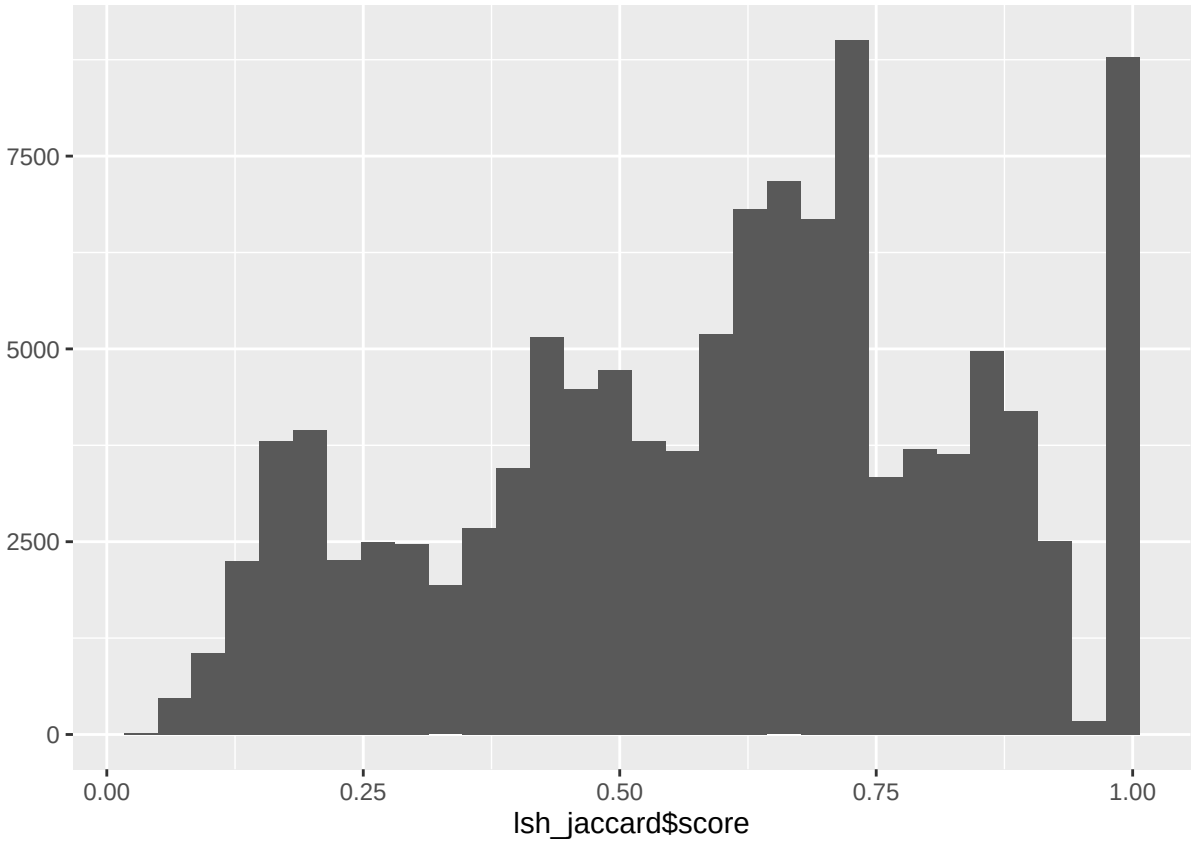
```
## [1] 2
```

```
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)

```
## `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).

```
N_total <- choose(n, 2)
N_lsh <- nrow(candidates)
reduction_ratio <- 1 - (N_lsh / N_total)
reduction_ratio
```

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

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.

Hint: the lsh_jaccard contains our pairwise predictions and cora_gold contains the pairwise truth. How can you compare the pairs in lsh_jaccard and the pairs in cora_gold to find the number of true positives first? Then, how can you extend this to find the other components of precision and recall? There are multiple ways to approach this problem, but examining the data pairwise may be the most intuitive.

```
# Format true pairs from cora_gold
true_pairs <- cora_gold[, c("id1", "id2")]
true_pairs <- apply(true_pairs, 1, function(x) paste(sort(x), collapse = "_"))

# Format predicted pairs from lsh_jaccard
predicted_pairs <- as.data.frame(lsh_jaccard[, c("a", "b")])
predicted_pairs <- apply(predicted_pairs, 1, function(x) paste(sort(x), collapse = "_"))

true_positives <- sum(predicted_pairs %in% true_pairs)
precision <- true_positives / length(predicted_pairs)
recall <- true_positives / length(true_pairs)

precision
```

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

```
recall
```

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

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.

```
filtered_lsh_jaccard <- lsh_jaccard %>%
  filter(score >= 0.8)

filtered_pairs <- as.data.frame(filtered_lsh_jaccard[, c("a", "b")])
filtered_pairs <- apply(filtered_pairs, 1, function(x) paste(sort(x), collapse = "_"))
```

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

```r
filtered_true_positives <- sum(filtered_pairs %in% true_pairs)

filtered_precision <- filtered_true_positives / length(filtered_pairs)
filtered_recall <- filtered_true_positives / length(true_pairs)
filtered_reduction_ratio <- 1 - (nrow(filtered_lsh_jaccard) / N_total)

filtered_precision
```

```
## [1] 0.8024888
```

```r
filtered_recall
```

```
## [1] 0.3205581
```

```r
filtered_reduction_ratio
```

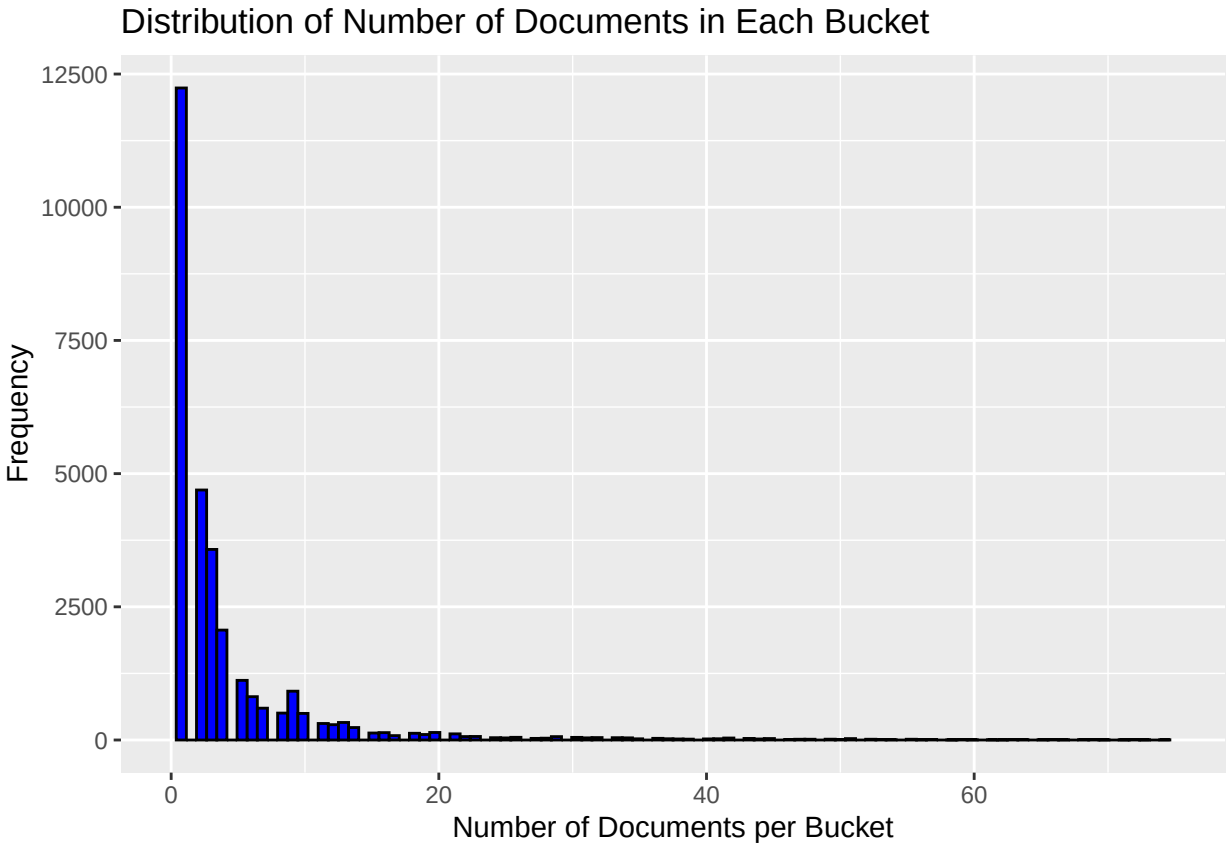
```
## [1] 0.9853796
```

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.

```r
bucket_counts <- buckets %>% group_by(buckets) %>% summarize(count = n())

ggplot(bucket_counts, aes(x = count)) +
  geom_histogram(bins = 100, fill = "blue", color = "black") +
  xlab("Number of Documents per Bucket") +
  ylab("Frequency") +
  ggtitle("Distribution of Number of Documents in Each Bucket") +
  xlim(0, 75)
```



The number of documents per bucket is skewed to the left indicating that there are many buckets that are underpopulated. Some documents are not finding many similar documents. This indicates that similar documents are not always being placed in the same buckets.

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

```
doc_bucket_counts <- buckets %>%
  group_by(doc) %>%
  summarise(bucket_count = n_distinct(buckets)) %>%
  filter(bucket_count > 1)

nrow(doc_bucket_counts)
```

```
## [1] 1879
```

Yes, there are overlapping blocks however, there are very few (~1%) blocks that overlap. Bands and rows can lead to overlapping buckets because a document might hash to the same value across different bands, resulting in it being placed in multiple buckets to increase the probability of it matching. However, according to the histogram above and the number of overlapping docs, it is clear that most docs are not being placed into multiple buckets and hence, there are many buckets with few records.

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

Advantages:

1. By reducing the number of comparisons needed (here, by 93%), LSH makes it more efficient to find similar records.
2. In addition to time efficiency, LSH also reduces the storage space of the data by bringing it into a single signature matrix.

Disadvantage:

1. Depending on the tuning parameters, LSH can fail to find many of the similar records, leading to poor performance.
2. There may be low recall and/or precision due to the approximation of results.