

# 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.

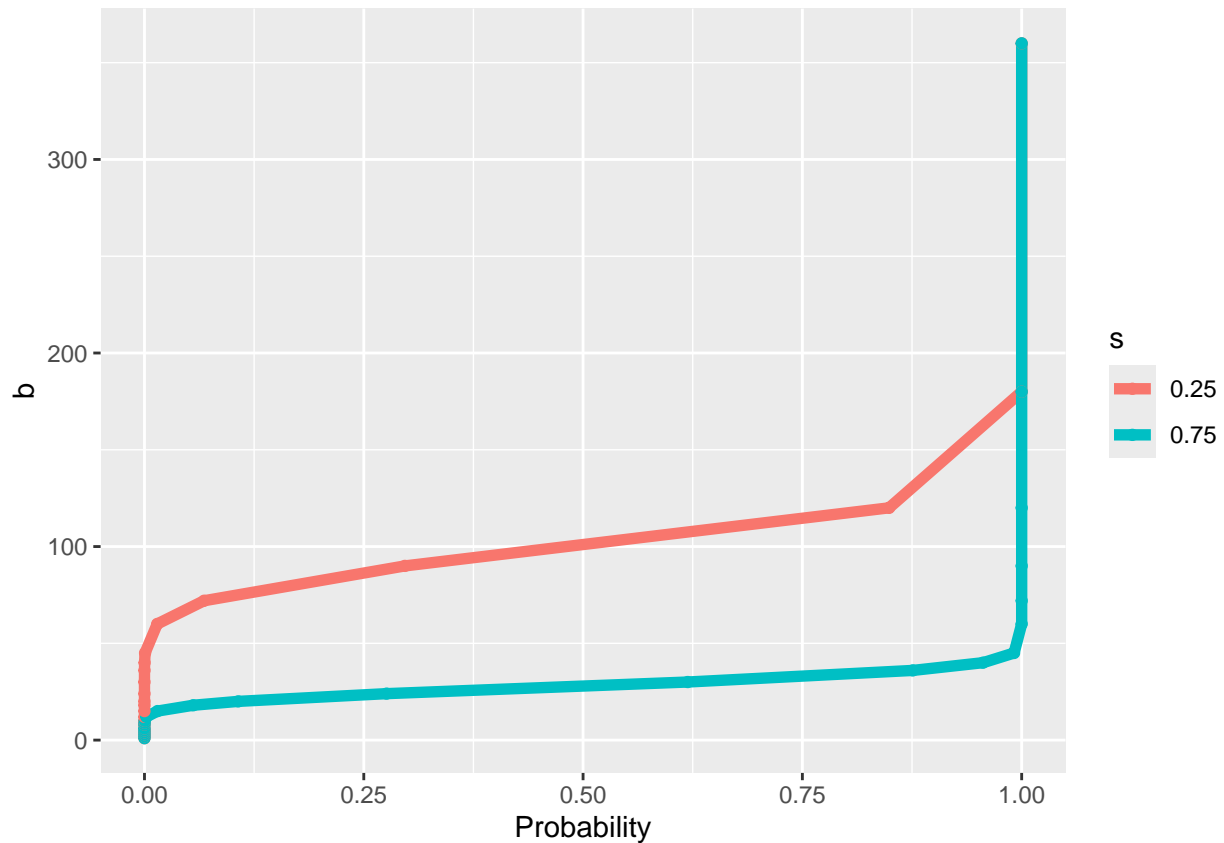
```
# get only the columns we want
# number of records
n <- nrow(cora)
# create id column
dat <- data.frame(id = seq_len(n))
# get columns we want
dat <- cbind(dat, cora[, c("title", "authors", "journal")])
```

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))
```

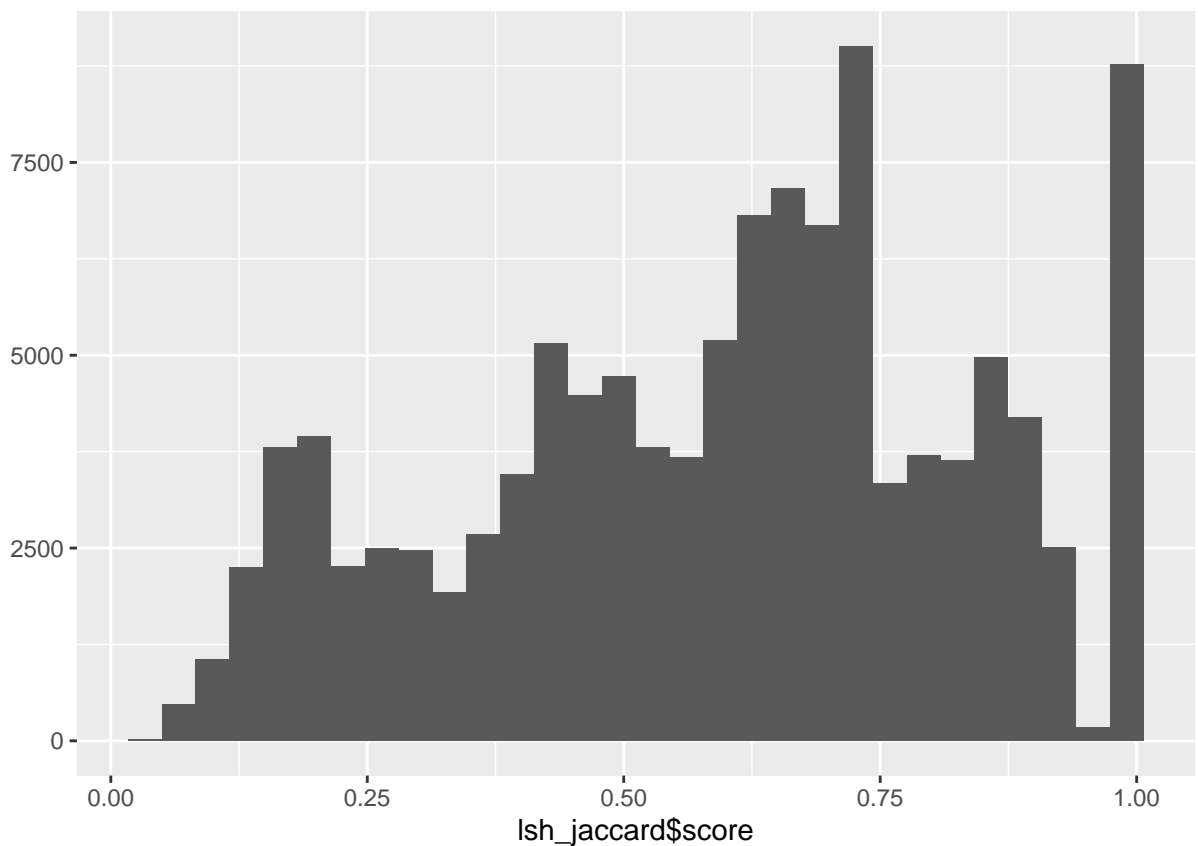
```
## [1] 2
```

```
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).

```
#count number of n
num_comparison <- nrow(candidates)
#calculate reduction ratio
(choose(n,2)-num_comparison)/choose(n,2)
```

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

Locality sensitive hashing reduces the comparison space by 93.48%.

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 identifier alternatively. You will need to write your own code for this.

```
calculate_TFPN <- function(candidates, a, b, truth,id1,id2){
  #Create pair for easy comparison
  pairs_candidates <- paste(candidates$a,candidates$b)
  pairs_truth <- paste(truth$id1,truth$id2)
  #True Positive = #incidents in candidates that are also in ground truth
  TP <- sum(pairs_candidates %in% pairs_truth)
  #False Positive = #incidents in candidates that are not in ground truth
  FP <- sum(!(pairs_candidates %in% pairs_truth))
  #False Negative = #incidents in ground truth but not in candidates
  FN <- sum(!(pairs_truth %in% pairs_candidates))

  precision <- TP/(TP+FP)
  recall <- (TP/(TP+FN))

  return(list(precision = precision, recall = recall))
}
#call function above
precision_and_recall <- calculate_TFPN(lsh_jaccard,a,b,cora_gold,id1,id2)
precision_and_recall$precision
```

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

```
precision_and_recall$recall
```

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

The precision says that 51.08% of record pairs that are classified as matches correspond to true matches, while the rest correspond to false matches.

The recall says that 90.87% of true matching record pairs are correctly classified as matches.

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.

```
#function for filtered jaccard score
filtered_by_threshold <-function(jscore,t){
  #keep candidates that have jaccard score greater than 0.8
  filtered_candidates <- subset(lsh_jaccard,score >=t)
  return(filtered_candidates)
}
filtered_candidates<-filtered_by_threshold(lsh_jaccard,t=0.8)
```

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

```
calculate_TFPN <- function(candidates, a, b, truth,id1,id2){
  #Create pair for easy comparison
  pairs_candidates <- paste(candidates$a,candidates$b)
  pairs_truth <- paste(truth$id1,truth$id2)
  #True Positive = #incidents in candidates that are also in ground truth
  TP <- sum(pairs_candidates %in% pairs_truth)
  #False Positive = #incidents in candidates that are not in ground truth
  FP <- sum(!(pairs_candidates %in% pairs_truth))
  #False Negative = #incidents in ground truth but not in candidates
  FN <- sum(!(pairs_truth %in% pairs_candidates))

  precision <- TP/(TP+FP)
  recall <- (TP/(TP+FN))

  return(list(precision = precision, recall = recall))
}

#call function and input parameters
precision_and_recall_filtered <- calculate_TFPN(filtered_candidates,a,b,cora_gold,id1,id2)
precision_and_recall_filtered$precision

## [1] 0.8024888
precision_and_recall_filtered$recall

## [1] 0.3205581
#recalculate number of n
num_comparison_filtered <- nrow(filtered_candidates)
(choose(n,2)-num_comparison_filtered)/choose(n,2)

## [1] 0.9853796
```

The precision says that 80.25% of record pairs that are classified as matches correspond to true matches, while the rest correspond to false matches.

The recall says that 32.05% of true matching record pairs are correctly classified as matches.

Locality sensitive hashing reduces the comparison space by 98.54%.

5.

- i. Describe what the blocks look like from this method? In the LSH Process, blocking happens as the signature matrix gets divide into bands, which further get hash into buckets.
- ii. Are they non-overlapping or overlapping? The blocks are overlapping because, theoretically, a single record can belong to multiple blocks, with other records sharing similarities across those blocks. This is evident from the fact that there are more unique buckets than the total number of records represented in both the graph and the count data below.

```
#count occurrence of buckets
occurrences<-table(buckets$buckets)
#sort the occurrences
sorted_occurrences <- sort(occurrences, decreasing = TRUE)

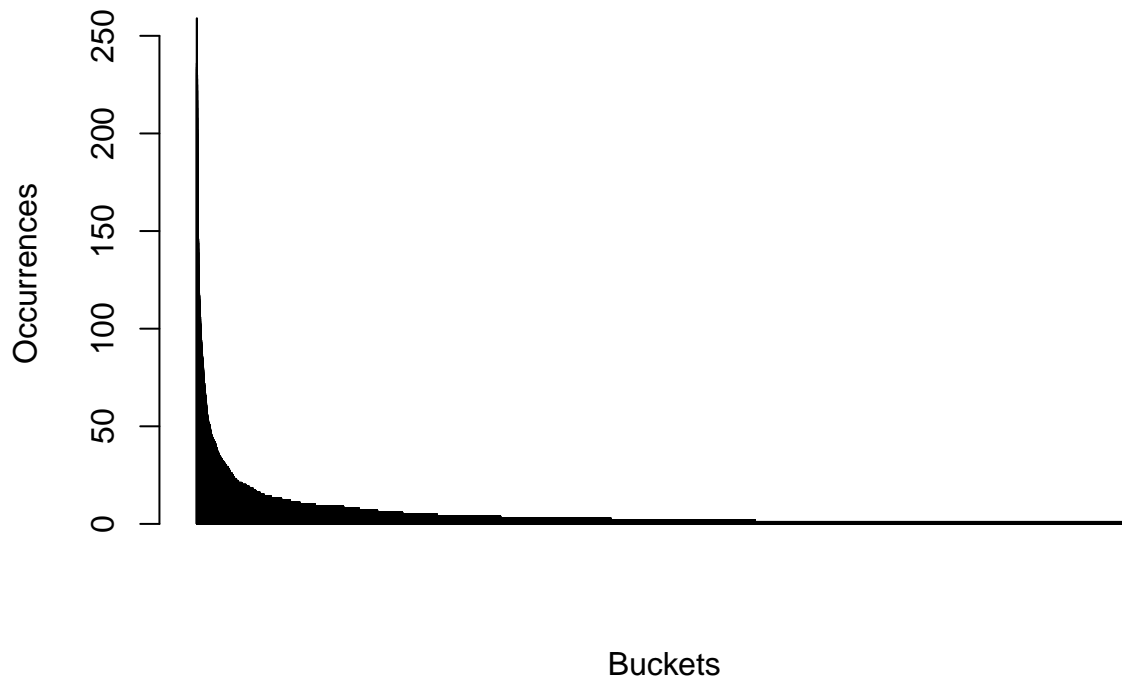
# Plot the barplot with sorted occurrences
barplot(sorted_occurrences,
        main = "Sorted Histogram of Occurrences", # Title of the plot
```

```

xlab = "Buckets",           # Label for the x-axis
ylab = "Occurrences",       # Label for the y-axis
col = "lightblue",          # Color of the bars
border = "black",           # Border color of the bars
xaxt = 'n')                 # Remove x-axis tick marks

```

## Sorted Histogram of Occurrences



```

# Get unique buckets (those with at least one data point)
unique_buckets <- unique(buckets$buckets)

# Count the number of unique buckets
length(unique_buckets)

```

```
## [1] 30355
```

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

Advantages: LSH is cost-efficient for large dataset as it reduces the comparison space significantly by 98.54% while maintains relatively high precision.

Disadvantages: LSH requires tuning for choosing number of bands and buckets. LSH also can lead to low recall because it utilizes probabilistic blocking.