STA 325L Lab (Sep 30)

Midterm Review

Athena Ru

Exam Instructions

- Exam I, Take Home Exam
- Release Date: Tuesday, October 1 at 10 AM.
- Due Date: Monday October 7th at 5 PM.

- Content: Modules 1 – 4 and homework 1 – 3.
- **The TA's will not be able to discuss exam format, etc.**
- There will not be lecture materials on Wednesday (Oct. 3) or Friday (Oct 5) to provide you with additional time to complete the exam or ask clarifying questions.

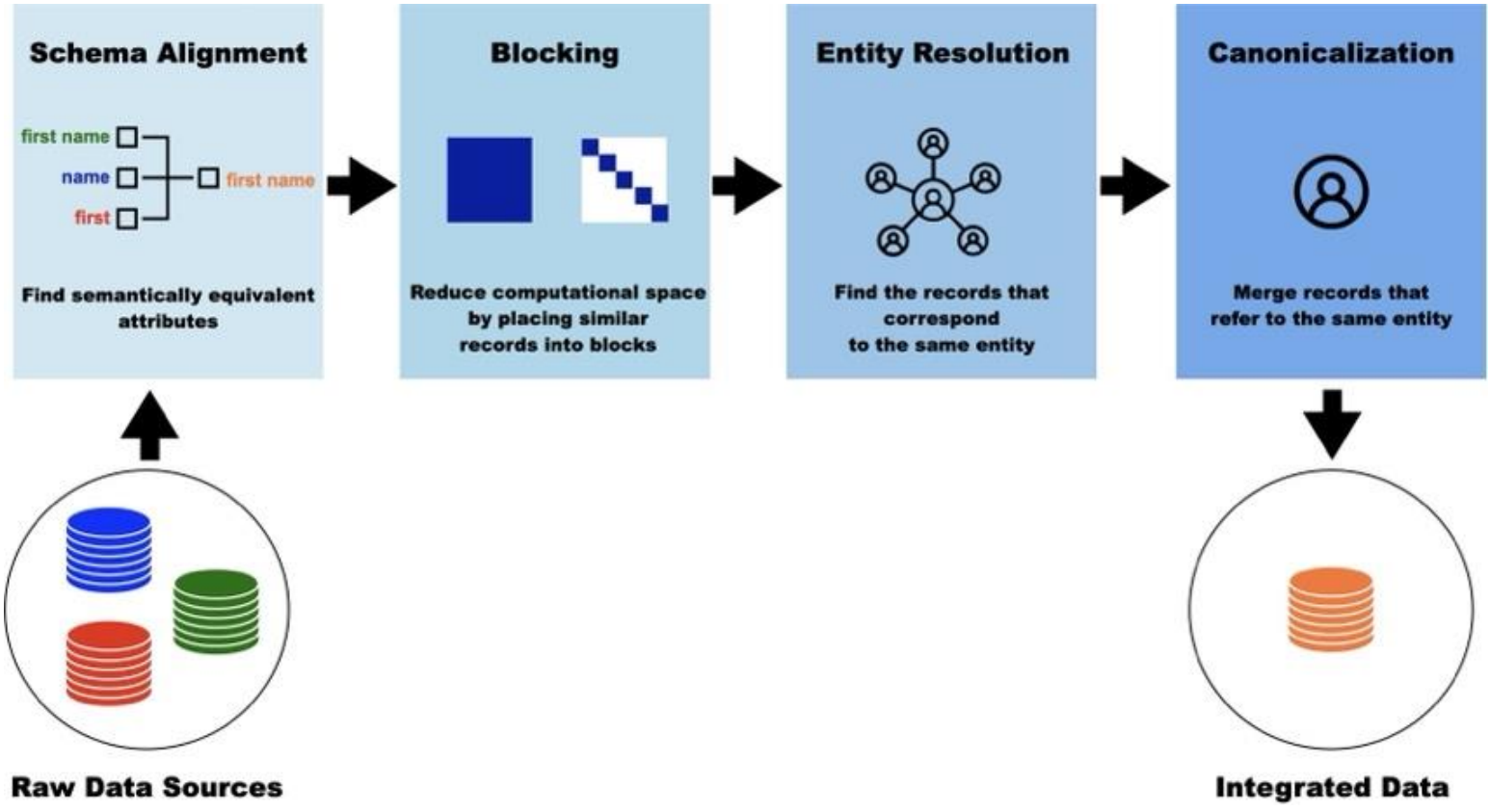
Exam Instructions Cont.

- This is a taken home examination to be completed individually.
- Open note, open book, and class resources (please cite, i.e., Module 4 Slide 13).
- The exam is closed to talking to students (or anyone else except your instructor or TAs in the course).
- We will only answer **clarification questions** and not anything about course material for the exam.
- You may not use Google, ChatGPT, or other online resources. You may utilize any suggested books or resources listed for the course materials.
- You may not talk to anyone about the exam until the grades are released to the entire class.
- Clarification questions will be addressed and should be asked to the instructor (and teaching assistants privately on Canvas).
- No group postings should be made regarding the exam or exam related material.
- Your submission must be to Gradescope and to Canvas (in the same format) as the homework.

Data Cleaning Pipeline

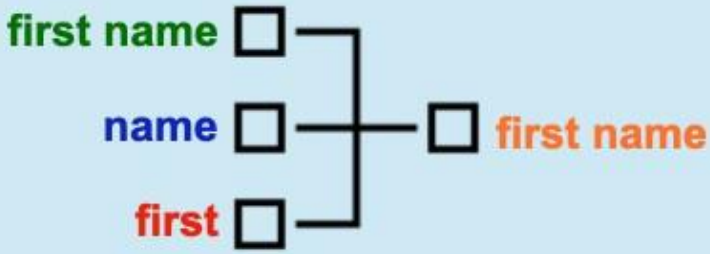
Rebecca C. Steorts

August 30, 2024



Recognize the 4 steps!

Schema Alignment



Find semantically equivalent attributes

Blocking

Reduce computational space by placing similar records into blocks

HW 2

Entity Resolution

Find the records that correspond to the same entity

HW 3

Canonicalization

Merge records that refer to the same entity

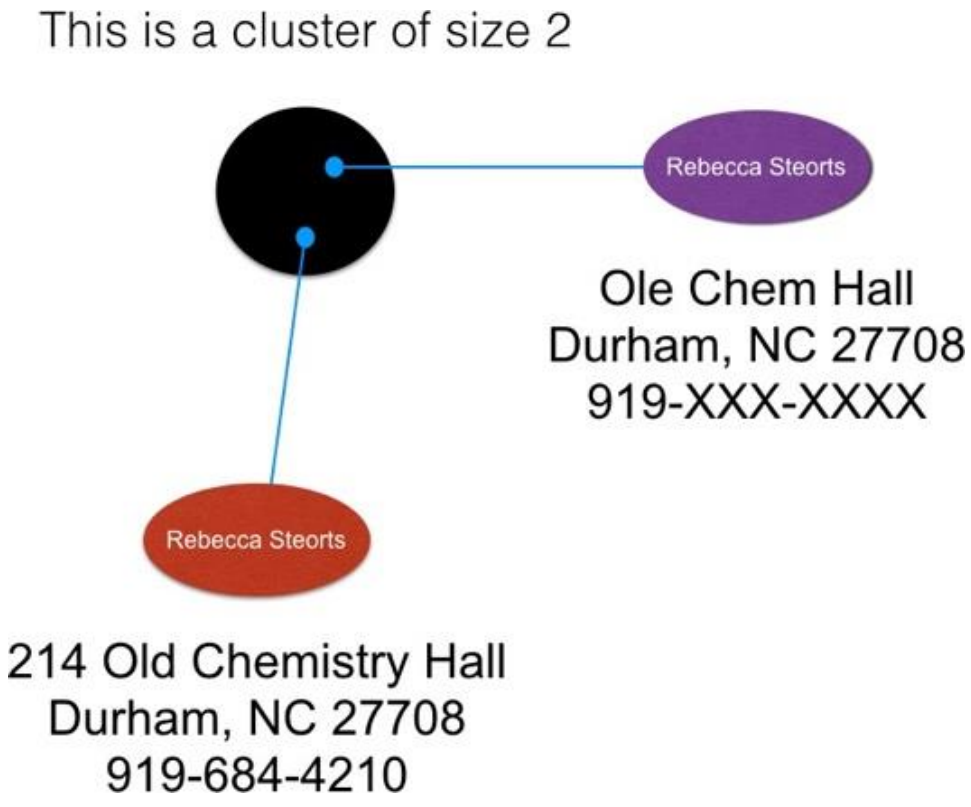
Preliminaries

Rebecca C. Steorts

September 6, 2024

Entity resolution (ER) is the process of merging together noisy (structured) databases to remove duplicate entities, often in the absence of a unique identifier.

Goal of Entity Resolution



! To find the most representative records after ER, one must perform **canonicalization** (data fusion or merging).

Challenges of Entity Resolution

Costly manual labelling

Vast amounts of manually-labelled data are typically required for supervised learning and evaluation.

Scalability/computational efficiency

Approximations are required to avoid quadratic scaling. Need to ensure impact on accuracy is minimal.

Limited treatment of uncertainty

Given inherent uncertainties, it's important to output predictions with confidence regions.

Unreliable evaluation

Standard evaluation methods return imprecise estimates of performance.

True Positive (True Matches)

True Positive (TP): These are records that are classified as matches and are true matches.

These are pairs of records that refer to the same person (entity).

H3 Q2: Pairs in *lsh_jaccard* that are also in cora_gold

True Negatives (True Non-Matches)

True Negative (TN): These record pairs are classified as non-matches, they are true non-matches.

The two records refer to two different entities.

Pairs NOT in *lsh_jaccard* that are also NOT in cora_gold

False Positive (False Matches)

False Positive (FP): These are record pairs that have been classified as matches, but they are not true matches.

The model made a wrong decision with the record pairs and falsely declared them to be matches.

H3 Q2: Pairs in *lsh_jaccard* that are NOT in cora_gold

False Negatives (False Non-Matches)

These are record pairs that have been classified as non-matches, but they are actually true matches.

The two record pairs refer to the same entity, but the method made a mistake.

H3 Q2: Pairs in *cora_gold* that are not in *lsh_jaccard*

Confusion Matrix

- Match (M)
- Non-Match (NM)
- N = total records

N		Predicted	
		M	NM
Actual	M	TP (true matches)	FN
	NM	FP	TN (true non-matches)

Review H2 Q3 (a)

Evaluation Metrics

$$\text{Precision} = \frac{TP}{TP + FP}$$

Measures how precise a method is in classifying true matches.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Measures how accurately the actual true matching pairs of records are correctly classified as matches.

Review H3 Q2

Evaluation Metrics

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Harmonic mean of the precision and recall.
- Attempts to summarize all aspects of the effectiveness of an entity resolution method.
- If low F-measure: in general, the classifier has difficulty accurately classifying record pairs correctly most of the time.

Review H2 Q3 (e)

Module 3: Deterministic Blocking

Rebecca C. Steorts

joint with Olivier Binette

Blocking

- Blocking places similar records into partitions/blocks.
- ER (typically) is only performed within each block.

Traditional Blocking

- Benefits: simple, easy to understand, and fast to implement.
- Downsides: the blocks are treated as error free, which is not usually accurate and can lead to errors in the ER task that cannot be accounted for.

Evaluation Metrics

Evaluation metrics are important for ER as they help us evaluate our proposed methodology (as long as some notion of ground truth exists).

We will focus on:

- reduction ratio
- precision
- recall
- f-measure

Reduction Ratio

The reduction ratio (RR) measures the relative reduction of the comparison space from the de-duplication or hashing technique.

See Christen (2012), Steorts, Ventura, Sadinle, Fienberg (2014) for a formal definition.

Pairwise evaluation metrics

$$\text{Recall} = \frac{TP}{TP + FN} = 1 - \text{FNR}.$$

$$\text{Precision} = \frac{TP}{TP + FP} = 1 - \text{FDR}.$$

$$\text{F-measure} = 2 \times \frac{(precision \times recall)}{(precision + recall)}.$$

Review H2 Q4 (d)

Recall

- For blocking, it is critical the recall be as close a possible to 1.
- To think about why, what does it mean if we have a blocking criterion where our recall is 0.5?

50 percent of true matching pairs are correctly classified as matches – no better than a coin toss.

RLdata500 (Continued)

```
# Total number of all to all record comparisons
choose(500,2)
```

```
## [1] 124750
```

Total all-to-all record comparisons = $\binom{n}{2}$

RLdata500 (Continued)

What is the overall dimension reduction from the original space to the reduced space induced by blocking?

How do we calculate the reduction ratio (RR)?

The reduction ratio is

$$\text{RR} = \%\ \text{comparisons eliminated by blocking.}$$

```
(choose(500, 2) - sum(choose(recordsPerBlock, 2))) /
choose(500, 2)
```

```
## [1] 0.8813226
```

How do we calculate the RR (via a function)?

```
reduction.ratio <- function(block.labels) {
  n_all_comp = choose(length(block.labels), 2)
  n_block_comp = sum(choose(table(block.labels), 2))

  (n_all_comp - n_block_comp) / n_all_comp
}

reduction.ratio(last_init)
```

```
## [1] 0.8813226
```

This function only works for blocking (you had to write your own in HW3 Q1)

Reduction Ratio

In summary, we have reduced the comparison space by roughly by 88 percent.

Pairwise Precision

```
precision <- function(block.labels, IDs) {
  ct = xtabs(~block.labels+IDs)

  # Number of true positives
  TP = sum(choose(ct, 2))

  # Number of positives = TP + FP
  P = sum(choose(rowSums(ct), 2))

  return(TP/P)
}
```

For blocking only!

Pairwise Recall

```
recall <- function(block.labels, IDs) {
  ct = xtabs(~IDs+block.labels)

  # Number of true positives
  TP = sum(choose(ct, 2))

  # Number of true links = TP + FN
  TL = sum(choose(rowSums(ct), 2))

  return(TP/TL)
}
```

For blocking only!

Interpretation

- The recall says that 100 percent of the true matching record pairs are correctly classified as matches.
- The precision says that 0.34 percent of record pairs that are classified as matches correspond to true matches, while the rest correspond to false matches.

In summary, we would not want to use this for an entity resolution algorithm. Why is this?

Summary

- Blocking is a method that puts similar records into blocks or bins.
- Main metrics for blocking are recall and reduction ratio.
- Balance between size of blocks and number of blocks.

Module 4: Probabilistic Blocking, Part I

Rebecca C. Steorts

LSH

Locality sensitive hashing (LSH) is a fast method of blocking for record linkage that orginates from the computer science literature.

Finding similar records

Our goal is to find *similar* records, where the records are assumed to be strings

How do we define *similar*?

Jaccard similarity

Consider the *Jaccard similarity*:

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

where

$|S \cap T|$ = the number of members shared between sets S, T

and

$|S \cup T|$ = the total number of members in both sets S, T.

How can you do this in R?

Jaccard similarity

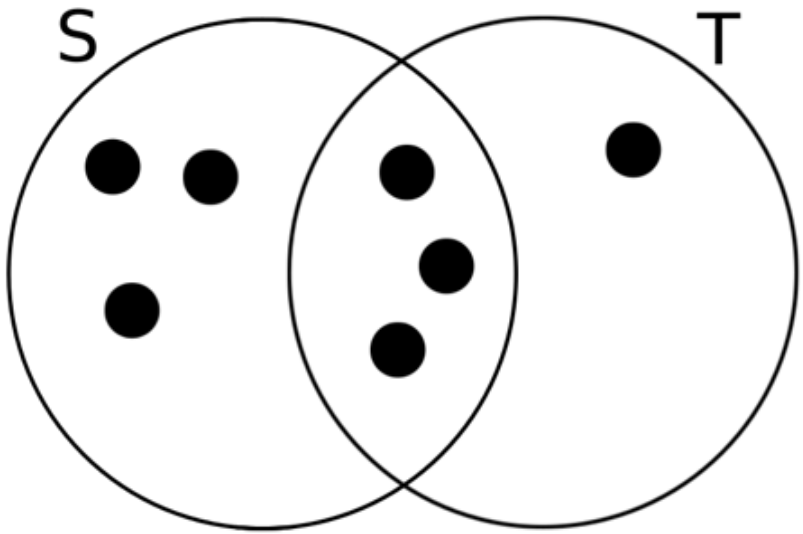


Figure 3: Two sets S and T with Jaccard similarity 3/7. The two sets share 3 elements in common, and there are 7 elements in total.

How to represent data as sets?

We want to talk about the similarity of our data (records)$\Longrightarrow$ we need to compare sets of records!

- We can construct a set of **short strings** from the data
- This is useful because similar datasets will have many common elements (common short strings)
- We can do construct these short strings using *shingling*

k-shingling (how-to)

1. Think of the data set as a string of characters

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

3. Associate with each data set the set of k-shingles that appear one or more times

Let's try

Suppose our data set is the string "Hello world", 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 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 15 items total {mi, ic, ch, ha, ae, el, lv, vo, og, ge, lm, me, ey, ye, er},

so the Jaccard similarity is $\frac{6}{15} = \frac{2}{5} = 0.4$

3. You should have learned that this is very tedious to do by hand!

Useful packages/functions in R

From the exercise, you should have leamed that we don't want to do this by hand!

Here are some useful packages in R that can help us!

```
library(textreuse) # text reuse/document similarity
library(tokenizers) # shingles
```

All packages needed for the exam will be preloaded.

Your turn (solution)

```
# create shingles for both names
token.1 <-
  tokenize_character_shingles("MICHAELVOGEL", n=2)
token.2 <-
  tokenize_character_shingles("MICHAELMEYER", n=2)
# compute jaccard similarity
# use unlist() to flatten the list into a character vector
jaccard_similarity(unlist(token.1),unlist(token.2))
```

```
## [1] 0.4
```

Module 4: Probabilistic Blocking, Part II

Rebecca C. Steorts

LSH

Locality sensitive hashing (LSH) is a fast method of blocking for record linkage that orginates from the computer science literature.

Hash function

We are looking for a hash function $h()$ such that

- if $\text{sim}(A, B)$ is high, then with high prob. $h(A) = h(B)$.
- if $\text{sim}(A, B)$ is low, then with high prob. $h(A) \neq h(B)$.

Hashing shingles

1. Instead of storing the strings as shingles, we store the **hashed values**.
2. These are integers (instead of strings).

We do this because the integers take up less memory, so we are performing a type of **dimension reduction**.

Characteristic matrix (continued)

We can visualize the records (columns) and the hashed shingles in a large, binary **characteristic matrix**.

Characteristic matrix (continued)

```
# inspect results
kable(char_mat[1:10, 5:10])
```

	Record 5	Record 6	Record 7	Record 8	Record 9	Record 10
466016402	1	1	1	1	1	0
-105604907	1	1	1	1	1	0
1270983012	1	1	1	1	0	0
-1170519627	1	1	1	1	0	0
2082481628	1	1	1	1	0	0
60115271	1	1	1	1	0	0
-883068501	1	1	1	1	1	0
-1062212181	1	1	1	1	1	1
-	1	1	1	1	1	0

Similarity preserving summaries of sets

Sets of shingles are large (larger than the original data set)

If we have millions of records in our data set, it may not be possible to store all the shingle-sets in memory

We can replace large sets by smaller representations, called *signatures*

We can use the *signatures* to **approximate** Jaccard similarity

Apply the Minhash

We want to create the signature matrix through minhashing

1. Permute the rows of the characteristic matrix *m* times
2. Iterate over each column of the permuted matrix
3. Populate the signature matrix, row-wise, with the row index from the first 1 value found in the column

The signature matrix is a hashing of values from the permuted characteristic matrix and has one row for the number of permutations calculated (*m*), and a column for each record.

Minhash Example

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Minhash Example

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

Step 1: Permute rows

Minhash Example

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

Step 2 & 3: Find row index with first “1” in each column
First row of Signature Matrix: [3, 5, 1, 3]

Minhash Example

Repeat *m* times to create signature matrix.

- Minhash function basically permutes (re-orders) the rows of the characteristic matrix

Signature matrix and Jaccard similarity

The relationship between the Signature matrix and the Jaccard Similarity is

$$Pr\{\min[\pi(A)] = \min[\pi(B)]\} = \frac{|A \cap B|}{|A \cup B|}$$

Min-hashing two sets gives same value

- We use this relationship to **approximate** the similarity between any two records
- The number of agreements over the total number of combinations is an approximation to Jaccard measure

Locality Sensitive Hashing (LSH) to the Rescue

We want to hash items several times such that similar items are more likely to be hashed into the same bucket.

1. Divide the **signature matrix** into b bands with r rows each so $m = b * r$ where m is the number of times that we drew a permutation of the characteristic matrix in the process of minhashing
2. Each band is hashed to a bucket by comparing the minihash for those permutations
   - If they match within the band, then they will be hashed to the same bucket
3. If two documents are hashed to the same bucket, they will be considered candidate pairs

We only check *candidate pairs* for similarity

HW 3 Q2 Partial Solutions

```
# 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)
```

General tips

- Review HW (especially HW 3 partial solutions on GitHub)
- All packages needed will be loaded for you
- You do not need online sources to solve the exam – everything can be found in HW or course slides
- Do what the Hint tells you to do (you might think there's a quicker/easier way, but you will likely get a different number!)
- If you have a clarification question when taking the midterm, please post privately to Canvas
- Good luck!