Lecture 7: Record Linkage

COSC 480 Data Science, Spring 2017

Michael Hay

Goals for today

- Wrap up SQL
- Record linkage (aka fuzzy matching)

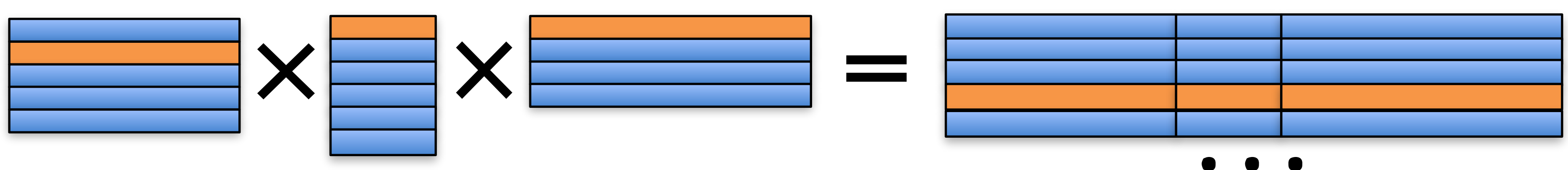
Putting it together

`SELECT` *columns or expressions*

4. Compute one output row for each "wide row" (or for each group of them if query has grouping/aggregation)

`FROM` *tables*

1. Generate all combinations of rows, one from each table; each combination forms a "wide row"

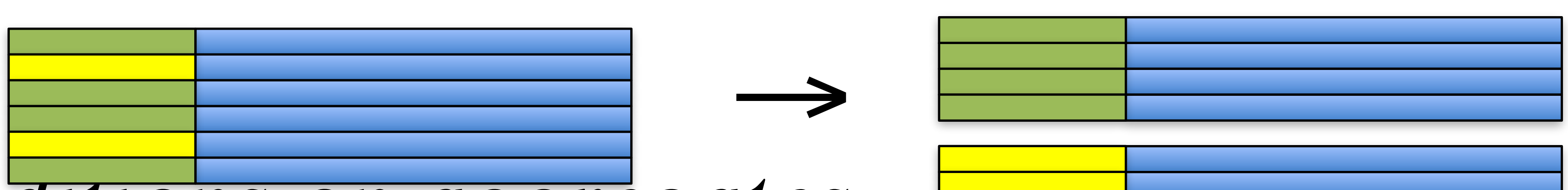


`WHERE` *conditions*

2. Filter—keep only "wide rows" satisfying *conditions*

`GROUP BY` *columns*

3. Group—"wide rows" with matching values for *columns* go into the same group



`HAVING` *conditions on aggregates*

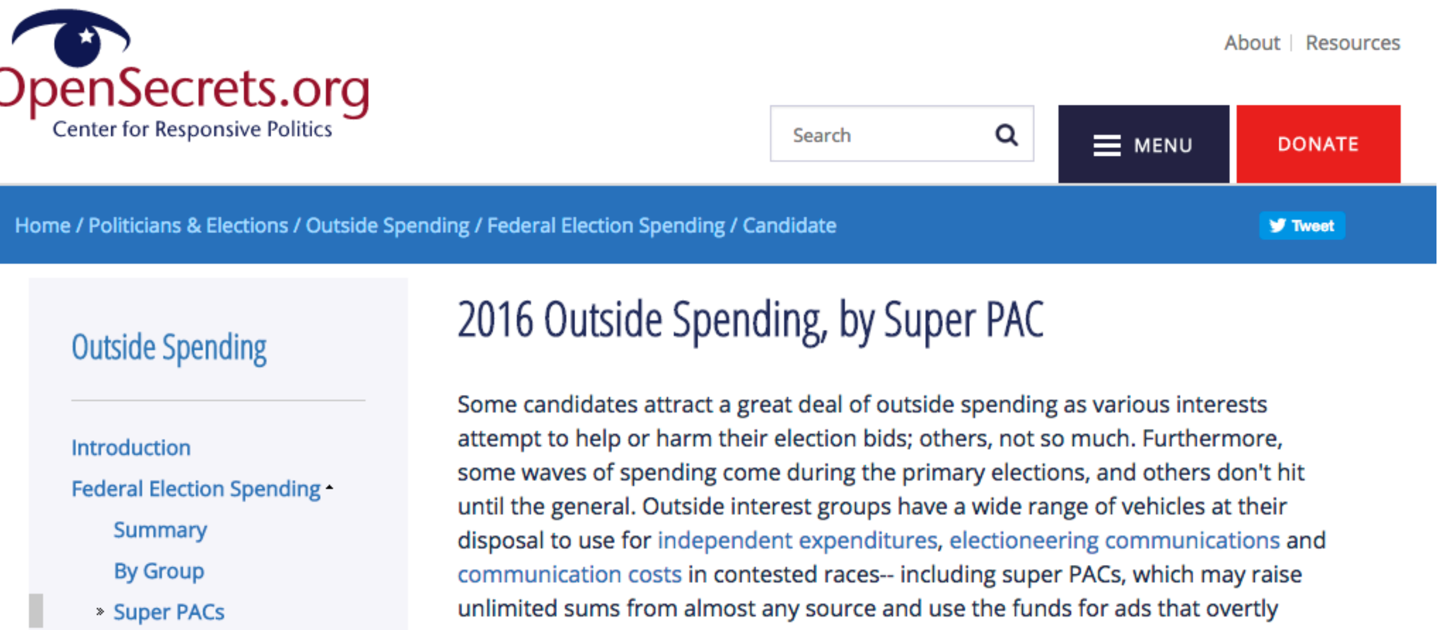
5. Filter groups based on aggregates

`ORDER BY` *output columns*;

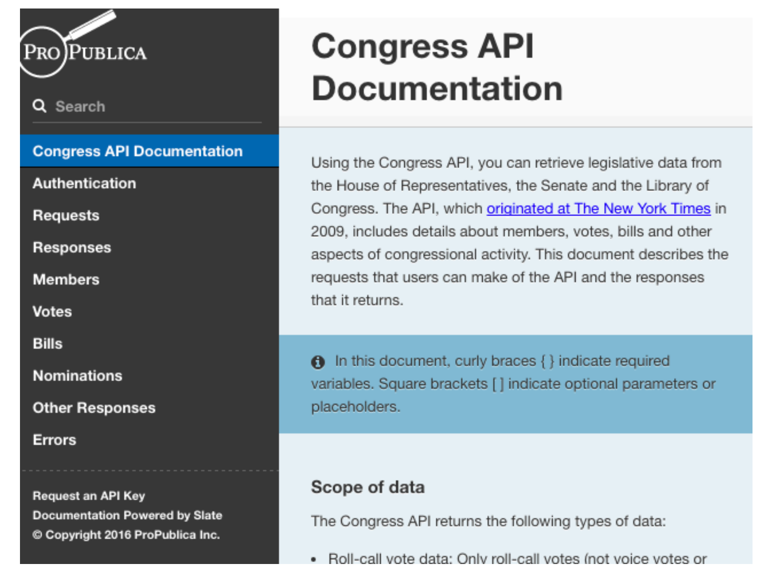
6. Sort the output rows

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Motivating example



https://www.opensecrets.org/outsidespending/summ.php



https://propublica.github.io/congress-api-docs/

- You have two different data sources, both describing the same set of entities (Congress members)
- You can put them both into the same relational database, but how do you join them? There is no key…

Record linkage

Record linkage

From Wikipedia, the free encyclopedia

Record linkage (RL) refers to the task of finding records in a data set that refer to the same entity across different data sources (e.g., data files, books, websites, databases). Record linkage is necessary when joining data sets based on entities that may or may not share a common identifier (e.g., database key, URI, National identification number), as may be the case due to differences in record shape, storage location, and/or curator style or preference. A data set that has undergone RL-oriented reconciliation may be referred to as being *cross-linked*. Record Linkage is called Data Linkage in many jurisdictions, but is the same process.

https://en.wikipedia.org/wiki/Record_linkage

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Ironically, record linkage has many names

Entity Resolution

Duplicate detection

Coreference resolution

Reference reconciliation

Fuzzy match

Object consolidation

Object identification

Deduplication

Entity clustering

Approximate match

Identity uncertainty

Household matching

Merge/purge

Hardening soft databases

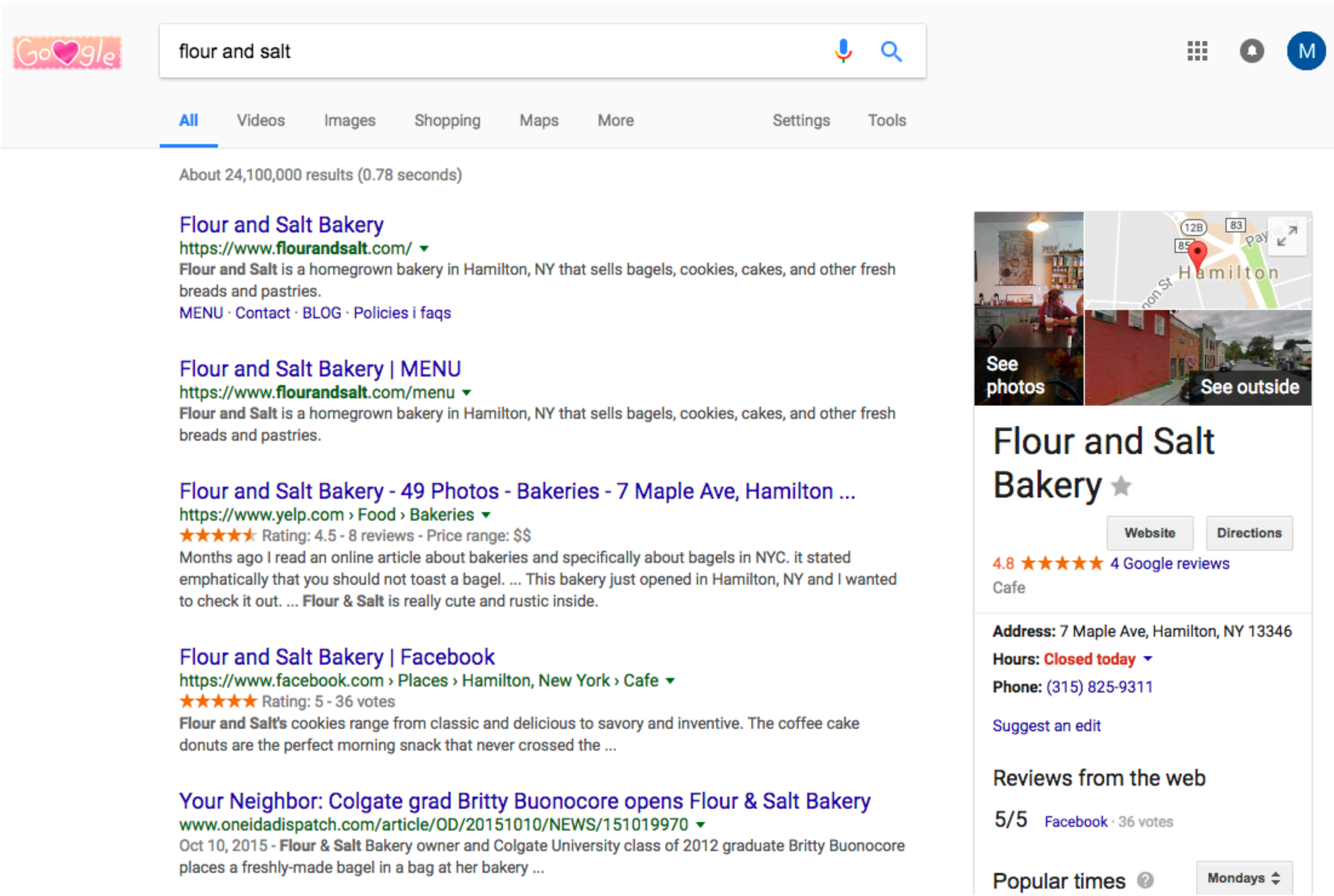
Reference matching

Householding

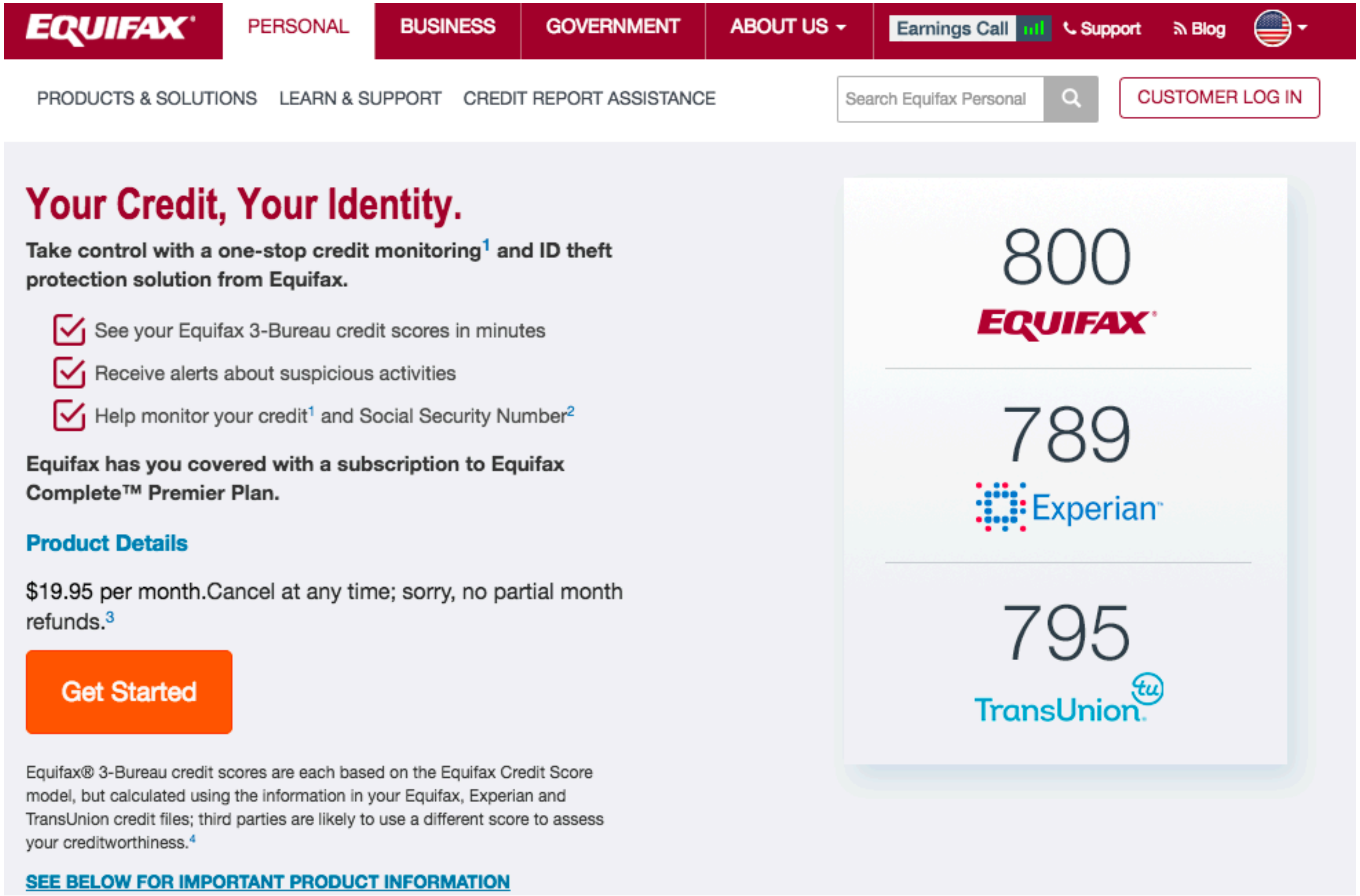
Doubles

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Motivating example: web



Motivating example: credit reports



Motivating example: networks

Measuring topology of internet using traceroute. IP aliasing problem:

```
$ traceroute google.com
traceroute to google.com (172.217.2.206), 64 hops max, 52 byte packets
 1  149.43.56.3 (149.43.56.3)  0.548 ms  0.341 ms  0.296 ms
 2  172.16.1.12 (172.16.1.12)  1.483 ms  1.323 ms  1.286 ms
 3  172.16.2.2 (172.16.2.2)  1.761 ms  1.480 ms  1.468 ms
 4  te0-4-0-9.ccr21.alb02.atlas.cogentco.com (38.104.52.97)  5.102 ms  4.9
 5  be2915.ccr41.jfk02.atlas.cogentco.com (154.54.40.62)  8.421 ms  8.348
 6  be2060.ccr21.jfk05.atlas.cogentco.com (154.54.31.10)  8.879 ms  8.312
 7  tata.jfk05.atlas.cogentco.com (154.54.12.18)  12.291 ms  12.172 ms  12
 8  if-ae-12-2.tcore1.n75-new-york.as6453.net (66.110.96.5)  12.460 ms  12
 9  72.14.218.224 (72.14.218.224)  12.741 ms  12.363 ms
    72.14.195.232 (72.14.195.232)  13.969 ms
10  216.239.50.106 (216.239.50.106)  13.266 ms
    209.85.248.242 (209.85.248.242)  14.117 ms
    216.239.62.127 (216.239.62.127)  12.980 ms
11  108.170.236.0 (108.170.236.0)  13.483 ms
    209.85.244.153 (209.85.244.153)  14.209 ms
    108.170.236.127 (108.170.236.127)  13.511 ms
12  108.177.3.59 (108.177.3.59)  19.432 ms  19.058 ms
    108.170.236.243 (108.170.236.243)  19.194 ms
13  216.239.48.94 (216.239.48.94)  18.952 ms
    108.170.235.156 (108.170.235.156)  18.729 ms  18.507 ms
14  72.14.233.91 (72.14.233.91)  20.096 ms  20.224 ms  19.335 ms
15  iad23s23-in-f206.1e100.net (172.217.2.206)  19.628 ms  18.866 ms  19.4
```

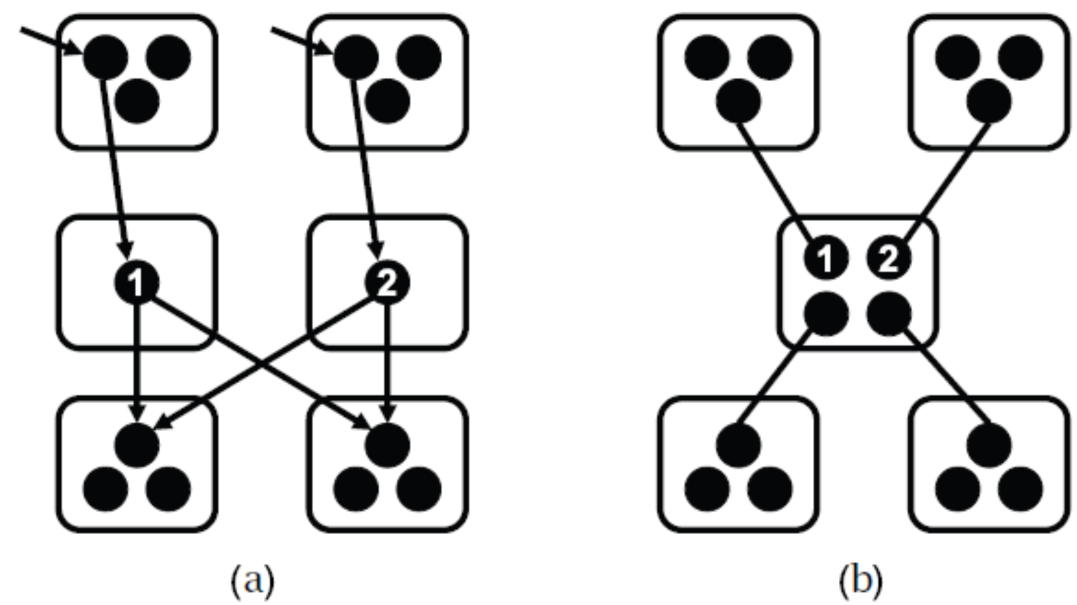


Figure 2. The IP alias resolution problem. Paraphrasing Fig. 4 of [50], traceroute does not list routers (boxes) along paths but IP addresses of input interfaces (circles), and alias resolution refers to the correct mapping of interfaces to routers to reveal the actual topology. In the case where interfaces 1 and 2 are aliases, (b) depicts the actual topology while (a) yields an "inflated" topology with more routers and links than the real one.

[Willinger et al. 2009]

Back to example

- How to link senator's records in two different data sources.
  - Join on (firstname, lastname)?
    - *Too specific* ("Joe" vs. "Joseph")
  - Join on just last name?
    - *Too inclusive* ("Smith")
  - Where is "Joe Liebermen"?
    - *Spelling mistakes, etc. Want approximate matching!*

```
Chris,Dodd,Democrat,CT,35.7,9161489
Richard,Shelby,Republican,AL,33.4,2542878
Charles,Schumer,Democrat,NY,32.8,3255362
Tom,Carper,Democrat,DE,32.5,1453446
Mike,Crapo,Republican,ID,32.2,946531
Bob,Bennett,Republican,UT,32.3,1078302
Jack,Reed,Democrat,RI,31.5,1280500
Tim,Johnson,Democrat,SD,29.1,1396308
Mike,Enzi,Republican,WY,25.1,564100
Joe,Liebermen,Independent,CT,25,7878838
```

Levenshtein (or edit) distance

- The minimum number of character `edit` operations needed to turn one string into the other.

LIEBERMAN
LIEBERMEN

- Substitute A to E. Edit distance = 1

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Levenshtein (or edit) distance

- Distance between two string *s* and *t* is the lowest cost sequence of edit commands that transform s to t.
- Edit commands
  - Copy character from *s* to *t*
  - Delete a character from *s*
    - Ex: *s* = "Joey" and *t* = "Joe"
  - Insert a character into *t*
    - Ex: *s* = "Hilary" and *t* = "Hillary"
  - Substitute one character for another
    - Ex: *s* = "Smyth" and *t* = "Smith"

	Cost
Copy	0
Delete	1
Insert	1
Sub	1

In general, costs could be different

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Example

s = `Joe Liebermen`

t = `Joseph Liberman`

Example

s = `Jo_e__ Liebermen`

ins del sub

t = `Joseph Li_berman`

Total cost: 3 + 1 + 1 = 5

Computing edit distance

- Two key observations

  1. We can contemplate edit distance between any substrings of *s* and *t*

     `cost(i,j)` = edit distance between `s[:i]` and `t[:j]`

Computing edit distance

t = "_Joey"

s = "_Joseph"

	_	J	o	e	y
_					
J					
o					
s					
e					
p					
h					

Cost of changing
_ ➡ _J

Cost of changing
_Jose ➡ _Jo

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Computing edit distance

- Two key observations

    1. We can contemplate edit distance between any substrings of *s* and *t*

       `cost(i,j)` = edit distance between `s[:i]` and `t[:j]`

    2. To compute cost(i,j), focus on effect of last edit command

Computing edit distance

t = "_Joey"

s = "_Joseph"

	_	J	o	e	y
_	0	1	2		
J	1	0	1		
o	2	1	0		
s	3	2	**1**		
e					
p					
h					

Cost of changing `_Jos` ➡ `_Jo.` Last edit command could be:

- Delete s: 1+ Cost(_Jo ➡ _Jo)
- Insert o: 1 + Cost(_Jos ➡ _J)
- Sub s with o: 1 + Cost(_Jo ➡ _J)

Set Cost(_Jos ➡ _Jo) to be the *minimum* of these options.

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Computing edit distance

t = "_Joey"

s = "_Joseph"

	_	J	o	e	y
_	0	1	2	3	4
J	1	0	1	2	3
o	2	1	0	1	2
s	3	2	1	1	2
e	4	3	2	1	2
p	5	4	3	2	2
h	6	5	4	3	3

Exercise

Instructions: ~work in groups of 2-3 to fill this in; then I will call on one of you

Compute edit distance between s = "_BCD" and t="_ABC". (The Joey example below is just for reference.)

t = "_Joey"

s = "_Joseph"

	_	J	o	e	y
_	0	1	2		
J	1	0	1		
o	2	1	0		
s	3	2	**1**		
e					
p					
h					

Cost of changing `_Jos` ➡ `_Jo.` Last edit command could be:

- Delete s: 1+ Cost(`_Jo` ➡ `_Jo`)
- Insert o: 1 + Cost(`_Jos` ➡ `_J`)
- Sub s with o: 1 + Cost(`_Jo` ➡ `_J`)

Set Cost(`_Jos` ➡ `_Jo`) to be the *minimum* of these options.

Computing edit distance

t = "_Joey"

s = "_Joseph"

	_	J	o	e	y
_	0	1	2	3	4
J	1	0	1	2	3
o	2	1	0	1	2
s	3	2	1	1	2
e	4	3	2	1	2
p	5	4	3	2	2
h	6	5	4	3	3

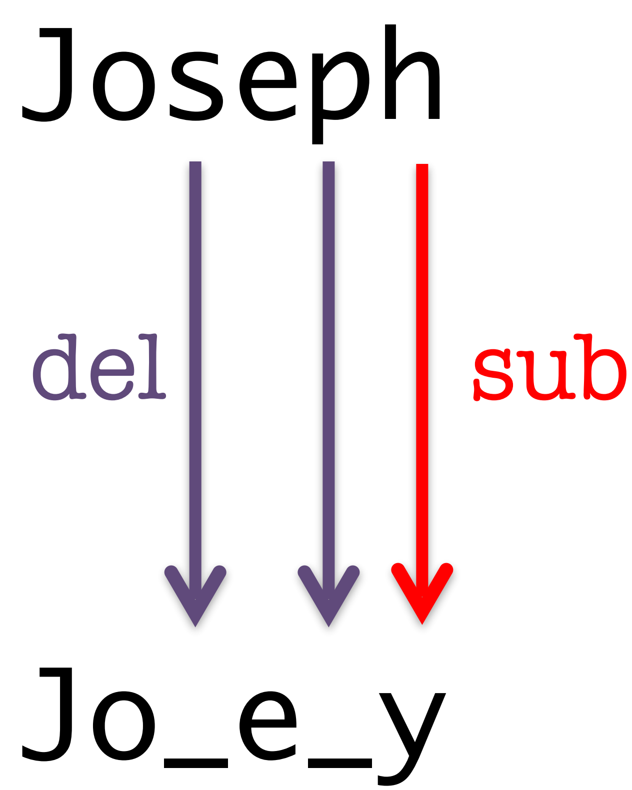
The edit distance between `Joseph` and `Joey` is 3, *but which edit commands achieve this*?

Computing edit distance

t = "_Joey"

s = "_Joseph"

	_	J	o	e	y
_	0	1	2	3	4
J	1	0	1	2	3
o	2	1	0	1	2
s	3	2	1	1	2
e	4	3	2	1	2
p	5	4	3	2	2
h	6	5	4	3	3



Remember the minimum in each step and retrace your path.

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Exercise

Instructions: ~1 minute to think/answer on your own; then discuss with neighbors; then I will call on one of you

Return to your previous example and note the minimum cost edit command at each step. (The Joey example below is just for reference.)

t = "_Joey"

s = "_Joseph"

	_	J	o	e	y
_	0	1	2		
J	1	0	1		
o	2	1	0		
s	3	2	**1**		
e					
p					
h					

Cost of changing `_Jos` ➡ `_Jo`. Last edit command could be:

- Delete s: 1+ Cost(`_Jo` ➡ `_Jo`)
- Insert o: 1 + Cost(`_Jos` ➡ `_J`)
- Sub s with o: 1 + Cost(`_Jo` ➡ `_J`)

Set Cost(`_Jos` ➡ `_Jo`) to be the *minimum* of these options.

Applying edit distance

- Back to motivating example: joining data about senators
- Some databases (e.g. postgresql) have built-in support for edit distance.
- Compute edit distance between firstname fields, and between last name fields
- Consider match if sum of distances below some *threshold*.
- Obviously, errors are possible: https://youtu.be/aRrDsbUdY_k?t=371

Edit distance variants

- Needleman-Wunsch
  - Different costs for each operation
- Affine gap penalty
  - "Joe Lieberman" vs. "Jo**seph I.** Lieberman"
  - Penalty for consecutive inserts: penalty for first insert + smaller penalty for each subsequent insert
- Edit distance has numerous applications (especially bioinformatics)

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Jaccard distance

- Distance function between two sets
    - *Can be applied to pairs of (long) strings too*
- Let A and B be two sets
    - Ex: words in two documents, friends lists of two individuals

$$\boxed{\mathrm{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}}$$

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Applied to names

- Use character *trigrams*

  LIEBERMAN = {__L, _LI, LIE, IEB, EBE,

  BER, ERM, RMA,MAN, AN_, N__}

  LIEBERMEN = {__L, _LI, LIE, IEB, EBE,

  BER, ERM, RME, MEN, EN_, N__}

- Jaccard(s, t) = 8/14 = 0.57

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Summary of Similarity Methods

Easiest and most efficient

- Equality on a boolean predicate
- Edit distance
  - Levenshtein, Affine
- Set similarity
  - Jaccard
- Vector Based
  - Cosine similarity, TFIDF
- Translation-based
- Numeric distance between values
- Phonetic Similarity
  - Soundex, Metaphone
- Other
  - Jaro-Winkler, Soft-TFIDF, Monge-Elkan

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

Summary of Similarity Methods

Handle Typographical errors

- Equality on a boolean predicate
- Edit distance
  - Levenstein, Affine
- Set similarity
  - Jaccard
- Vector Based
  - Cosine similarity, TFIDF

Good for Text (reviews/ tweets), sets, class membership, …

Useful for abbreviations, alternate names.

- Translation-based
- Numeric distance between values
- Phonetic Similarity
  - Soundex, Metaphone
- Other
  - Jaro-Winkler, Soft-TFIDF, Monge-Elkan

Good for Names

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

The Ugly side of Record Linkage
[Sweeney IJUFKS 2002]

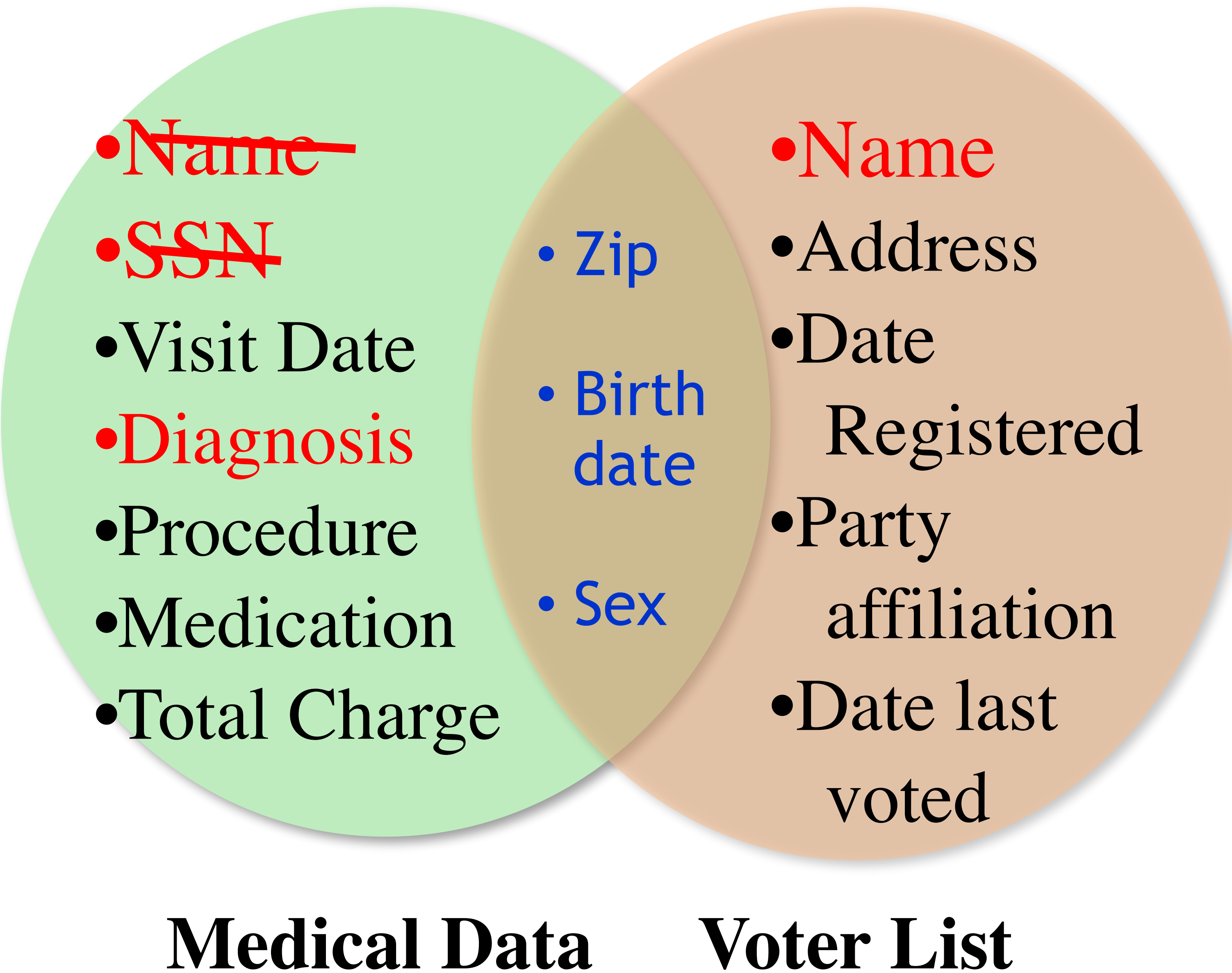
- ~~Name~~
- ~~SSN~~
- Visit Date
- Diagnosis
- Procedure
- Medication
- Total Charge

- Zip
- Birth date
- Sex

Medical Data

The Ugly side of Record Linkage

[Sweeney IJUFKS 2002]



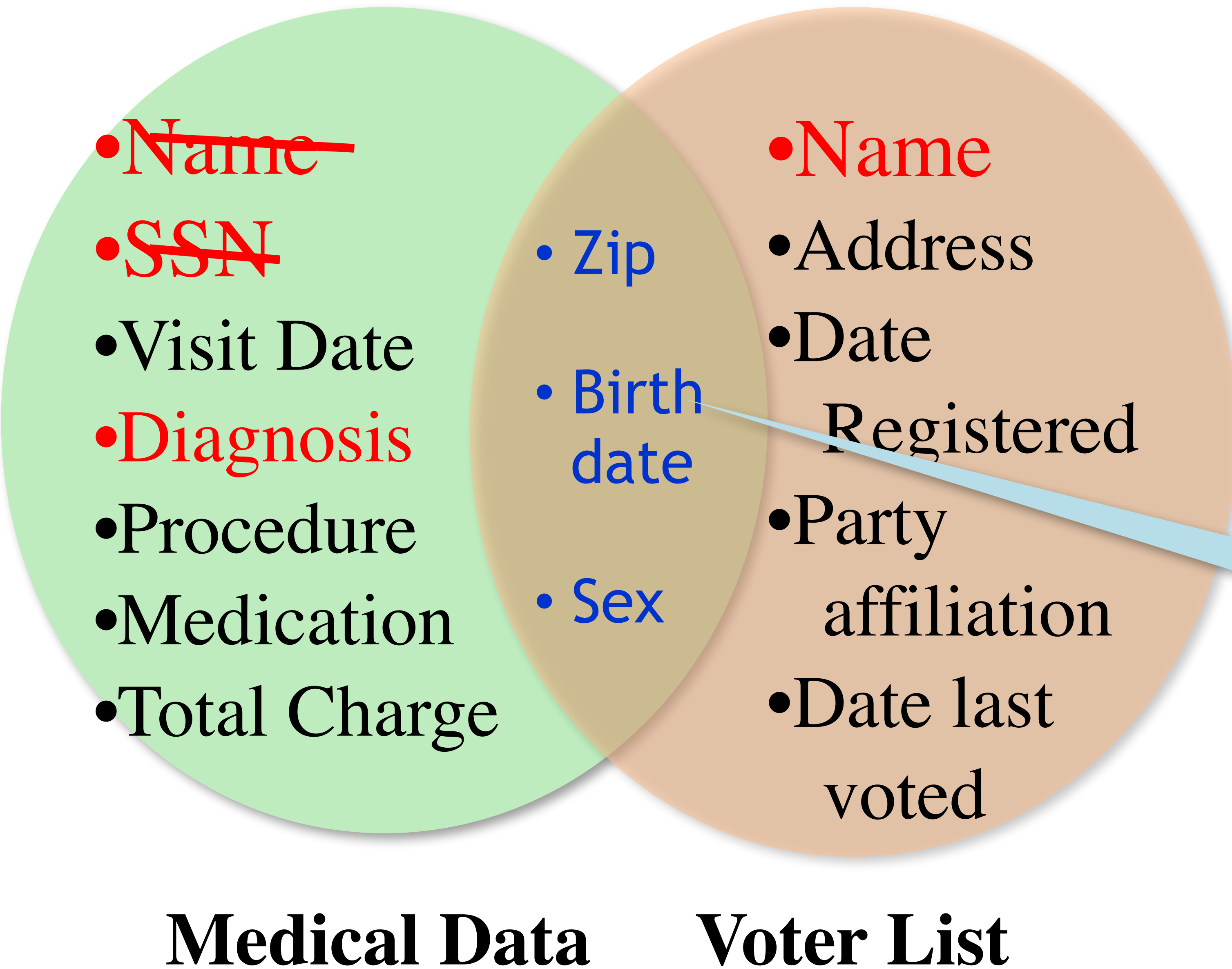
- Governor of MA **uniquely identified** using ZipCode, Birth Date, and Sex.

Name linked to Diagnosis

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University

The Ugly side of Record Linkage

[Sweeney IJUFKS 2002]



- 87 % of US population **uniquely identified** using ZipCode, Birth Date, and Sex.

Quasi Identifier

Credit: slide adapted with permission from Ashwin Machanavajjhalla and Jun Yang, Duke University