COMP20008 Elements of Data Processing

Semester 2 2018

Lecture 15: Data Linkage and Privacy - Bloom Filters

Announcements

- Project Phase 3 was released on Monday 10th September.
- Next lecture on Friday 14th September will be from Prof Richard Sinnott and is intended to get you thinking about what scenario to use for project phase3 and highlight an easy to use data repository
- Consultation sessions about Project Phase 2
  - Fri 14/09/2018 Room 09.02 Doug McDonell 1:00pm-2:00pm
  - Wed 19/09/2018 Room 07.02 Doug McDonell 10:30am-11:30am
- Workshop-week8-answers will be released tonight!

Phase 1 Assignment: General Feedback (1)

- Stats:
  - average mark: 16.5 , median: 17.5
- Data Cleaning:
  - Computation of missing values in each column was mostly done mostly correctly.
  - Some students processed date-time column as string.
  - When doing data cleaning, it is important to keep in mind that missing values may come in different forms, and not all of them are represented as NaN. One should apply domain knowledge and common sense to consider missing values.
- Visualisation:
  - Mostly students were familiar with plots and did the correct analysis
  - The justifications provided for plots were also appropriate but when making justifications, every conclusion needs to be based on underlying data.

Phase 1 Assignment: General Feedback (2)

- Visualisation:
  - A common mistake in Q2.3 was to provide that the median is more robust to outliers than mean and therefore the former must be used to impute missing values. In this particular dataset, there are outliers on both sides of the revenue, but the mean and median are practically indistinguishable so both can be used to impute missing values.
- Coding:
  - Codes were commented and documented properly.
  - In some cases, students used excessive loops. More compact/efficient code was expected!
- Justification:
  - Many struggled to make the right conclusion from the parallel coordinate plot, mainly because the plot was not correctly displayed.

Plan today

- A privacy preserving method for data linkage using approximate matching
  - Extension of method discussed last lecture, to allow approximate matching in a private manner

Summary/ Privacy preserving linkage

- Organisations A and B can determine which records in the two databases are an exact match in a privacy preserving manner by
  - using a trusted third party C, and
  - using one way hash functions with a salt, and
  - adding random records
- A reasonably private scheme (depending on how much the third party is trust)

Challenge

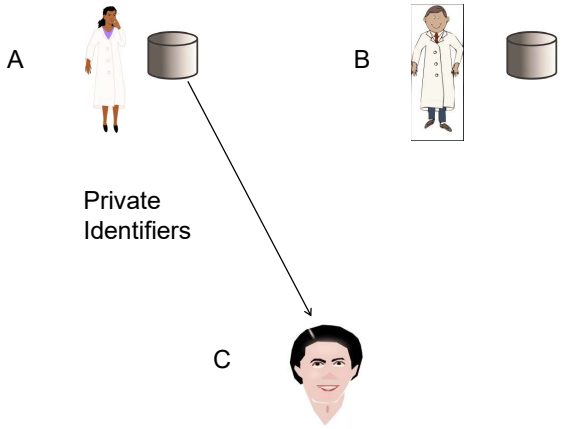
- The hash based technique using the 3rd party, can only compute exact similarity between strings in a privacy preserving manner.
- What if we wish to compute approximate similarity between two strings in a privacy preserving manner?
  - E.g. May want to match "James Bailey Unimelb" with "James Bailey Unimel". Hash outputs are completely different:
    - H(*James Bailey Unimelb*)
      - d1bd13e8442f345a6eeb903573b6cc4e5942111cb78dced8240ef9d4a09faf9faa20bc7754d15966e2d61f37a0f4a65d414fed8a8f4dd3be0989e90d05248b1b
    - H(*James Bailey Unimel*)
      - b7f060dbc73f7111eaba9488f78c096f687df04487d7fe40ffa2e5ae6b7b81a6897d3dfa7b4a3e1b28eaf6ad1503d403b726ff1af9a5996e4eddd507723b3a15

Desired Process

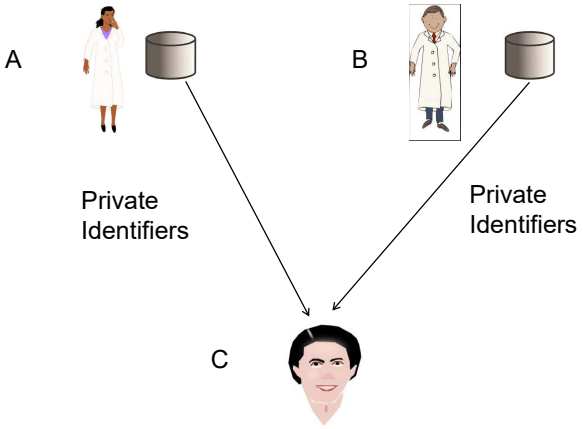
C

Trusted Party

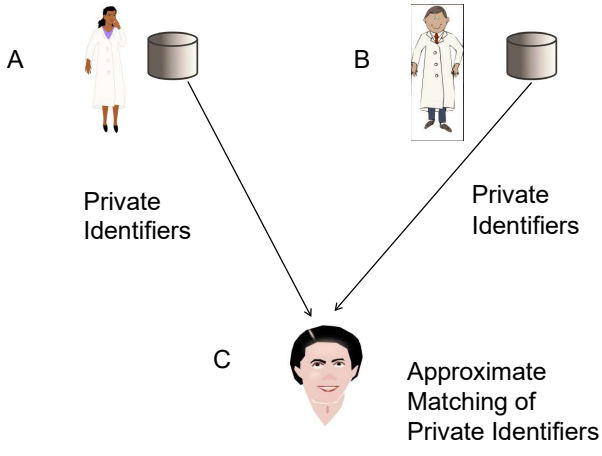
THE UNIVERSITY OF MELBOURNE | Intended Process



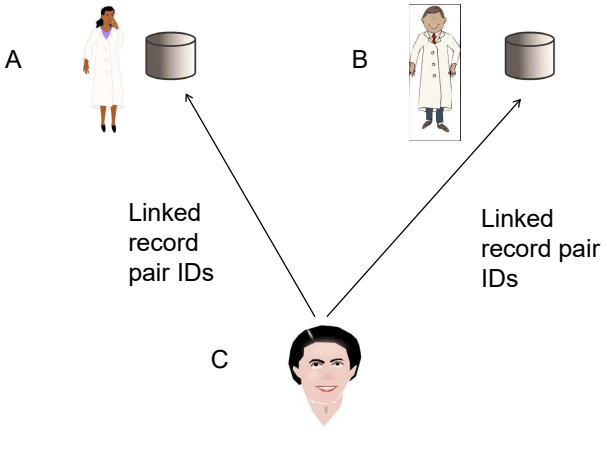
THE UNIVERSITY OF MELBOURNE | Intended Process



THE UNIVERSITY OF MELBOURNE | Intended Process



THE UNIVERSITY OF MELBOURNE | Intended Process



Components that are needed

1. Computing approximate similarity of two record fields (strings)
2. Representing a record field (string) in a privacy preserving manner (so that it is difficult to reverse engineer its value)
3. Computing approximate similarity of two record fields (strings) that have been represented in privacy preserving manner.

Issue 1: Approximate Similarity

- We wish to compute the (approximate) similarity between two strings X and Y, where $0 \leq sim(X,Y) \leq 1$
- Convert each string into 2-grams. Convert to lower case and list all substrings of length 2. E.g.
  - James -> ['ja', 'am', 'me', 'es']
  - Jamie -> ['ja', 'am', 'mi', 'ie']

$$sim(X,Y) = \frac{2h}{x+y}$$

  - Where
    - h= number of common 2-grams
    - x= number of 2 grams in string X
    - y= number of 2 grams in string Y
    - Known as the "Dice coefficient"
- $Sim(James, Jamie) = \frac{(2*2)}{(4+4)} = 0.5$

Exercise

- *Using 2-grams, what is the similarity between strings x and y, where*
  - *x=Melbourne*
  - *y=Bournemouth*
- *me el lb* ***bo ou ur rn ne***
- ***bo ou ur rn ne*** *em mo* ***ou*** *ut th*
- *h=5, x= 8 y=10*
- $Sim(x,y) = \frac{2*5}{8+10}$

Why 2-grams?

- Similarity measure can easily be extended to 3-grams, 4-grams, ….(q-grams)
- *Why not 1-grams?*
  - *COMP20008 versus 80002PMOC*
- *Why not 10-grams?*
  - *Biased against short words (not sensitive)*

Terminology

- We will be using a data structure called a “bit string” or a “bit array”
  - An array of binary digits (bits): 0’s and 1’s. E.g.

1	0	1	1	0	0	1	0

Issue 2: Bloom Filters

- A bloom filter is an array of l bits, with all bits initially set to zero. E.g. A bloom filter with l=25 bits will initially contain 25 zeros

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- We may store strings in the bloom filter by using hash functions $H_1 \ldots H_k$ to turn on certain bits.
  - Each hash function H_i ($1 \le i \le k$) maps an input string X to a value in the range [0,l-1].
  - To store a string X in the bloom filter, set array index $H_i(X)$ in the bloom filter to value ‘1’, for each $H_i(X)$

Storing strings in a bloom filter (E.g. If k=2, l=25) happy

- **To store the string “ha”**
- Suppose $H_1(ha)=22$, $H_2(ha)=15$
  - » We therefore set bits 22 and 15 to have value ‘1’ in the bloom filter
- **To store the string “ap”**
- Suppose $H_1(ap)=3$, $H_2(ap)=11$
  - » We therefore sets bits 3 and 11 to have value’1’ in the bloom filter
- **To store the string “pp”**
- Suppose $H_1(pp)=1$, $H_2(pp)=7$
  - » We therefore sets bits 1 and 7 to have value ‘1’ in the bloom filter
- **To store the string “py”**
- Suppose $H_1(py)=13$, $H_2(py)=15$
  - » We therefore set bits 13 and 15 to have value ‘1’ in the bloom filter

L=25 bits: inserting 2-grams of string Happy

Initially all zeros

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

↓ Insert “ha”, turn on bits 15 and 22

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0

↓ Insert “ap”, turn on bits 3 and 11

0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0

↓ Insert “pp”, turn on bits 1 and 7

0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0

↓ Insert “py”, turn on 13 (15 already on)

0	1	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	0

Bloom Filter: Storing a Set of Strings

- To store a set of strings $\{X_1, X_2, \ldots, X_n\}$ in the bloom filter, each element X_j is hash coded using the k hash functions and all bits having indices $H_i(X_j)$ are set to 1 (for $1 \leq i \leq k$ and for $1 \leq j \leq n$).
- If a bit was set to 1 before, no change is made.

Bloom Filter: Checking if a string is present

- For any string X, we can thus check if it is stored in the bloom filter by hashing X using the k hash functions.
  - If at least one of the bits having value $H_i(X)$ is set to 0, then X is definitely not a member of the bloom filter.
  - If all of the bits having value $H_i(X)$ are set to 1, then X appears to be a member of the bloom filter. However, it might not really be a member (a false positive)
  - *Question: Why isn't X a member of the bloom filter with certainty, if all bits having value $H_i(X)$ are set to 1?*

Bloom filters: A different Application

- Google Chrome Browser used to use bloom filters to store malicious URLs.
  - When you visited a website the browser checked if that URL is in the filter. If the answer was (possibly) yes, then the browser would query Google's servers for more detailed information. Otherwise, browsing would continue as normal.
  - The bloom filter was a compressed data structure of all the malicious URLs Google knew about, stored by the browser.
    - Much less space than maintaining a full database of URLs

Bloom filter: Comparing two strings for (approximate) similarity

- Given two strings
  - _SMITH_
    - 2-Grams: _S, SM, MI, IT, TH, H_
  - _SMYTH_
    - 2-Grams: _S, SM, MY,YT,TH, H_
- The 2-grams of the first string are stored in bloom filter B1, the 2-grams of the second string are stored in bloom filter B2. Both bloom filters are the same length and use the same hash functions.
- If the two strings have a lot of 2-grams in common, then their bloom filters will have a large number of identical bit positions set to 1.

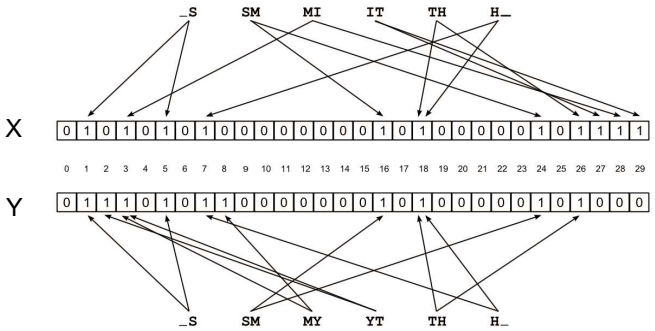
Issue 3: Similarity of bloom filters

- Use the formula

$$sim(B1, B2) = \frac{2h}{b1 + b2}$$

  - Where h is the number of bits set to 1 in both bloom filters
  - b1 is the number of bits set to 1 in bloom filter B1
  - b2 is the number of bits set to 1 in bloom filter B2

Similarity between _SMITH_ and _SMYTH_



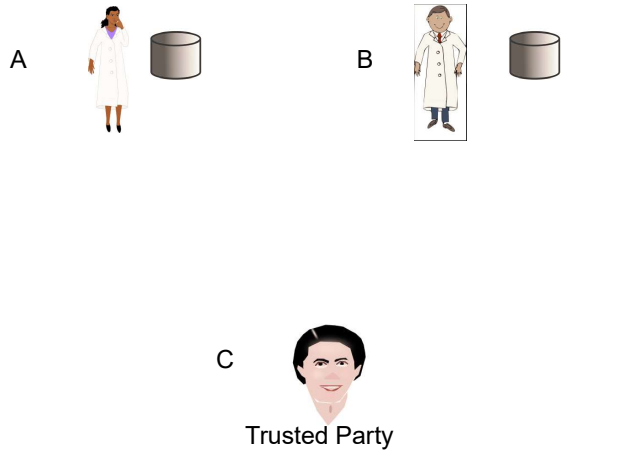
- String X has 11 bits set to 1. String Y has 10 bits set to 1
- There are 8 common bits set to 1.
- Sim(X,Y)= $\frac{2*8}{10+11} = 0.762$

- Given two strings X and Y and a similarity value SIM(X,Y)
  - Need to choose a threshold value for deciding whether to report that X matches Y (whether X is approximately similar to Y)

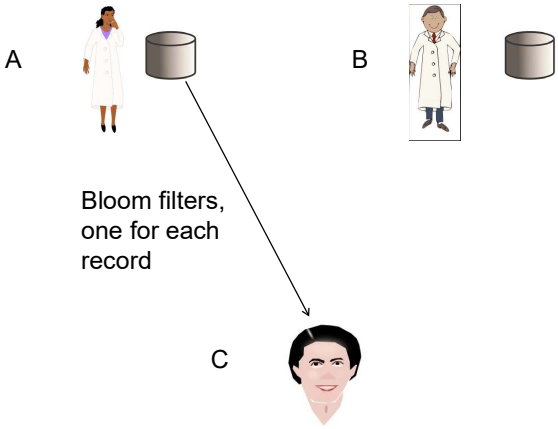
Performance of this methods

- Following work in "Privacy Preserving Record Linkage Using Bloom Filters", *BMC Medical Informatics and Decision Making*, Schnell et al, 2009
  - Two databases, 15,000 people each
  - Bloom filter with 2-grams
  - Bloom filter length l=500 bits
  - Number of hash functions k=15
  - Performance of bloom filter approach
    - Was almost exactly the same as approximate match using original string values (no bloom filter).
      - Identified almost the same matched pairs and non-matched pairs
- Parameter setting is important for performance and security

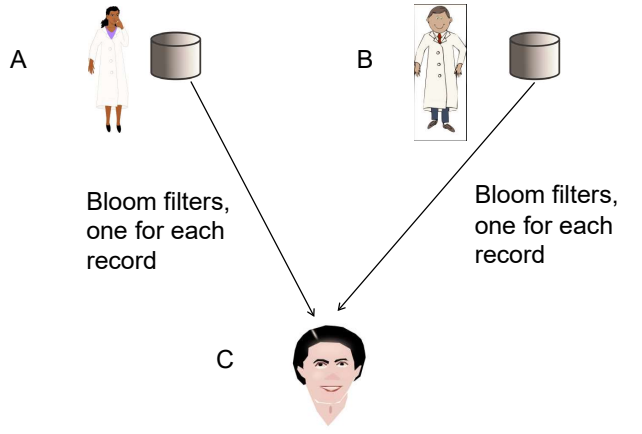
Privacy preserving (approximate) linkage process



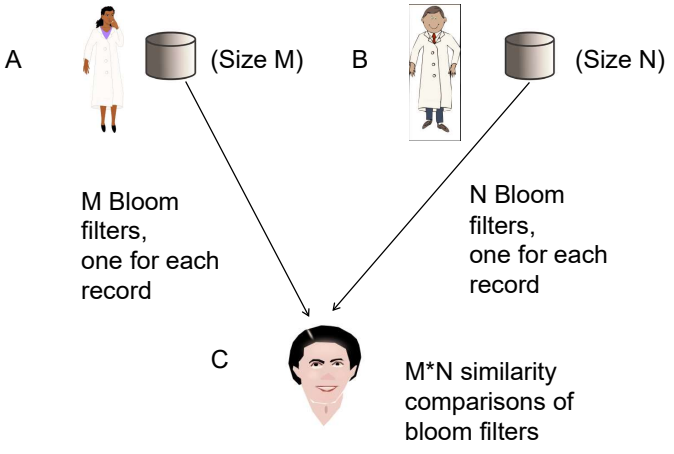
Privacy preserving (approximate) linkage process



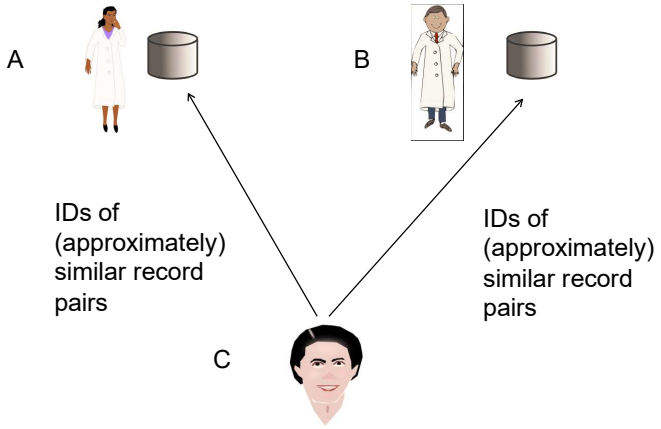
Privacy preserving (approximate) linkage process



Privacy preserving (approximate) linkage process



Privacy preserving (approximate) linkage process



Usage considerations

- A and B agree on a bit array length l and k hash functions
  - A and B also agree on a salt and use it in their hash functions
- A and B may also add dummy records (and their corresponding bloom filters) to lessen the chance of C mounting a frequency attack
- May be problems with using bloom filters to represent short strings (not very much secure)
  - A and B might add random 2-grams or random bits to the bloom filters, for extra security

Sample exam questions

- Given the strings wrangling and wrapping, compute their (approximate) similarity using 2-grams. Show all working
- Suppose a 14 bit bloom filter is used as a privacy preserving representation for strings. The bloom filter representation of wrangling is 11000010001111 and the bloom filter representation for wrapping is 11100001101111. What is the approximate similarity of these strings using the bloom filter representation?
  - 11000010001111
  - 11100001101111
  - $Sim = \frac{2*6}{7+9}$
- Explain how bloom filters can help provide a private representation for strings, in the context of privacy preserving data linkage.
  - Explain a bit about what bloom filter is and how word gets represented by bloom filter
  - Non invertible

Remaining Challenge: Public release

- Suppose organisation wishes to make one of its internal datasets public, for social good purposes
  - E.g. NASA releasing images of Mars
  - City of San Francisco, crime data
  - CERN, particle physics data
  - Bank, data on credit scoring and people who experiences financial distress
- Can be very, very difficult to prevent data linkage attacks or reverse engineering of people's identities
  - America Online search logs
  - Medicare Benefits Schedule data (https://pursuit.unimelb.edu.au/articles/understanding-the-maths-is-crucial-for-protecting-privacy)

America Online Search Logs

- In 2006, America Online released a file with 3 months of ``anonymized'' search queries of 658k users.
  - After a public outcry, data quickly taken down, but couldn't be removed completely from the Web
  - Ranked 58 out of the 101 dumbest moments in business by CNNMoney.com
  - http://www.nytimes.com/2006/08/09/technology/09aol.html?_r=0

User id	Time	Search Query
1	..	..
1	…	…
1	…	…
2	…	.
2	…	…
2	..	…
3	…	…

Acknowledgements

- Relevant work on data linkage, including privacy
  - Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution and Duplicate Detection, Peter Christen, Springer, 2012. Available as an e-book for download by University Library
    - Read Sections 1,2, 4.1,4.2,5.4, 8.1, 8.2
- This presentation closely follows the following paper
  - Privacy Preserving Record Linkage Using Bloom Filters, *BMC Medical Informatics and Decision Making*, Schnell et al, 2009
    - http://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/1472-6947-9-41

What you need to know from today

- understand the steps of the 3 party protocol for privacy preserving data linkage with approximate matching (using bloom filters)
- understand how to compute similarity of two strings based on 2-grams and why this method is useful
- understand how a bloom filter works (how it is created, how strings are inserted, how strings are checked for membership)
- understand how a bloom filter provides a private representation for strings
  -understand how bloom filters can be used to compare two strings for approximate similarity and the formula for doing this (Dice similarity coefficient)