Attempt to Reconstruct Confidential Data Using Public Data

Sarah Birmingham

Coding It Forward Data Science Fellow

Census of Fatal Occupational Injuries

August 5th 2021

Supervisor:

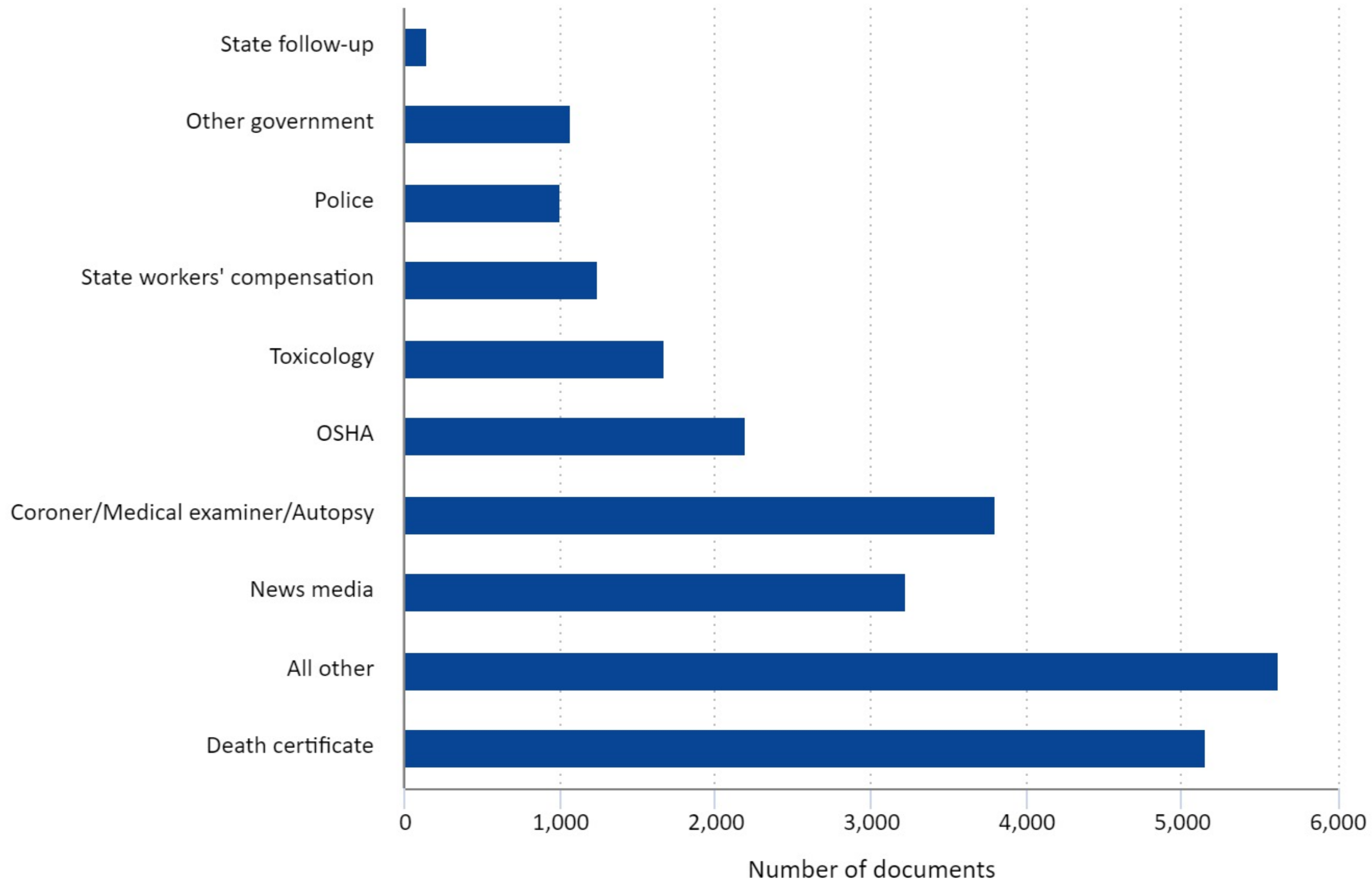
Ellen Galantucci, PhD

Mathematical Statistician

What is CFOI?

- The Census of Fatal Occupational Injuries is a count of fatal work injuries in the United States and measures the number of people who die at work or due to a work-related injury
- CFOI data is used by OSHA and NIOSH for regulatory and advisory purposes
- Data is obtained from a variety of sources and BLS is obligated to keep identifiable information confidential under CIPSEA

Sources of data on fatal work injuries, 2019



Hover over chart to view data.
Source: U.S. Bureau of Labor Statistics.

What are database reconstruction attacks (DRA)?

- Published data is used to infer underlying microdata
- Researchers at UT-Austin used the Netflix Prize dataset and public information from IMDB to identify Netflix records of known users

- CFOI publishes decedent data on:
  - Type, occupation, industry, and sector of employment
  - Gender, age, and race and Hispanic origin status
  - Causal event and state of occurrence
- Data is available on a national and per-state basis
- Need to balance usefulness of published data with responsibility to protect confidentiality
- Publishability requirements were changed in 2019
- Data is hard to protect because counts are small

Problem Set-Up

- Access to both public and confidential microdata; able to start with all known public data
  - Unrealistic for an actual adversarial attack
- Extract assignment rules from published tabular data
  - Rule examples include:
    - Four gender_2 in occupation_53
    - No government employees were self-employed
    - If not gender_1, gender_2

Record ID	State	Event	Occupation	Industry	Public/ private	Wage/ self-employed	Gender	Age category	Race
1	42	1	47	238	50	2	1	4	1
2	42	2	53	926	10	4	1	4	2
3	42	5	47	485	50	4	1	5	6
4	42	4	11	238	50	2	2	6	3
5	42	3	53	561	50	4	1	3	1
6	42	3	49	713	50	4	2	4	1
7	42	6	31	621	50	4	1	4	2

Initial Approach: Backtracking

- Conceptualized the problem as an enormous Sudoku puzzle or the game “Clue”: starting from known information, make initial inferences and eliminate other possibilities
- Assign values until I reach a solution or a dead end; in which case, backtrack

Record ID	State	Event	Occupation	Industry	Public/ private	Wage/ self-employed	Gender	Age category	Race
1	42	*1*	*11*	238	*50*	2	*1*	*5*	*3*
2	42	2			10				
3	42	5			50			5	6
4	42	4		238	50			6	
5	42	3							
6	42			713			2		
7	42		31						

Rule 1: Only one race_3
Rule 2: Occupation_11 has race_3
Rule 3: Race_3 has gender_2

Did it work?
No

- Problem size was way too big for this to be a feasible approach, even on individual states
- Sudoku on a board with > 50,000 cells
- Tried to modify this attempt by:
  - Assigning values pre-approved pairs
  - Reordering columns to fill in easier-to-guess cells first

Second Approach: Optimization with OR-Tools library

- Google OR-Tools is an open-source software suite for optimization problems
- Create a model by assigning variables and constraints
- Use a SAT solver (built-in, open-source, or commercial) to solve

Record ID	State	Event_1	Event_2	Event_3	Event_4	Event_5	Event_6	Event_?
1	42	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)
2	42	0	1	0	0	0	0	0
3	42	0	0	0	0	1	0	0
4	42	0	0	0	1	0	0	0
5	42	0	0	1	0	0	0	0
6	42	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)
7	42	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)

Did it work?
No

- Found one potential solution; couldn’t find optimal or multiple solutions
- Didn’t seem designed to handle problems that had an inconsistent number of assignable values
- No counting function

Third Approach: Merging Permutations

- Use tabular data to find # and value of assignable options
- Necessarily have some unknown values-will never be able to solve for those
- Create a list of valid permutations for each category, keeping public values fixed

Third Approach: Merging Permutations

- For each grouping of two categories [I,J] for which a ‘rule’ exists, eliminate members of both categories that don’t satisfy that rule
  - No rule between categories -> all pairs are valid
- For each grouping of two pairs [I,J],[J,K] that share a common category, eliminate members that don’t:
  - Satisfy any rules that exist between all three [i,j,k]
  - Have [i,k] in list of pairs [I,K]

	Event	Industry	Public/ private	Wage/ self-employed
	3	238	50	2
	2	426	10	4
	5	985	50	4
	4	238	50	4
	3	565	50	4
	5	713	50	4
	6	621	50	4
n perm	**6**	**24**	**1**	**5**

Total
[illegible]

Did it work?
Maybe!

- Continue to merge until you have a group of all nine categories
  - Ideally would yield one solution, but could result in many
  - I didn't see the problem space getting any smaller, and it was taking an hour+ for RI, which had 10 records
  - Generating list of permutations is time/memory expensive

Final Attempt

- Combination of previous attempts
- Generate one unique permutation/category at a time and assign to solution dataframe
- Check for consistency; if a category option can’t be assigned, backtrack to previous category

Record ID	State	Event	Occupation	Industry	Public/ private	Wage/ self-employed	Gender	Age category	Race
1	42	1	40	236		2			
2	42	2	53	926	10				
3	42	5	11	561	50			5	6
4	42	4	47	236	50			6	
5	42	3	53	485					
6	42	3	47	713			2		
7	42	6	31	621					

Did it work?
Maybe!

- Produces a consistent table for small states relatively quickly; accuracy varies
- Can “stack” results to check for values that are the same between results and fix in place
- Can’t predict how many accurate solutions
- Can’t eliminate impossible permutation options
- Without original microdata to check, difficult to determine correctness
- Still slow for larger states

Results

	Solved	Acc.	Speed	Pros	Cons
Backtracking	X		X	• Easy to understand	• Slow
OR-Tools Optimization	✓	X	✓	• Built-in functions	• Inflexible
Merging Permutations	X		Okay	• Eliminate large numbers quickly	• Slow overall • Uses lots of memory • Can't make per-value judgement
Combination	✓	Okay	✓	• Fastest • Multiple solutions • Look for consistency within variables	• Can't eliminate possibilities that will never work

Going Forward

- Combine state results to check for consistency on a national level
- Test with data published following rule changes

Citations

- A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 2008, pp. 111-125, doi: 10.1109/SP.2008.33.
- S. Garfinkel, Abowd J.M, and Martindale C. 2018. "Understanding Reconstruction Attacks on Public Data," *Communications of the ACM,* 2018, vol. 62 no. 3, pp 46-43, doi: 10.1145/3287287
- Google OR-Tools, Google Developers.

Citations

Contact Information

Sarah Birmingham
Coding It Forward Data Science Fellow
Census of Fatal Occupational Injuries
610-906-6593
sarahbirmingham42@gmail.com