The Data Breach, Record Linkage, and Private Computation

Alexander C. Mueller

Saint Louis Machine Learning and Data Science Meetup
April 14, 2020

Who is the speaker?

30 second resume:

- an ancient metro Saint Louis townie
- grew up in University City
- University City High School
- B.A. Washington University (econ and math)
- Ph.D. University of Michigan (math)
- 5-ish years of private sector data science
- founded data privacy company Capnion

Find me on LinkedIn or email me acmueller@capnion.com

Slides posted on https://github.com/capnion/random

Agenda

Personally identifiable information (PII)

- is sometimes moving too much, other times too little...
- is increasingly diverse and fuzzily defined
- is driven by data privacy and quality concerns

Record linkage: Many data sources $\Rightarrow$ one record

- What is it? Who cares? Business perspective
- Technical details in Python

How to provide better privacy with Ghost PII

- Computations on encrypted data
- Example grouping of surnames

Data is Moving...

Grindr shares personal data with ad companies in violation of GDPR, complaint alleges

A Norwegian nonprofit has filed three complaints against the company

By Jon Porter | @JonPorty | Jan 14, 2020, 12:25pm EST

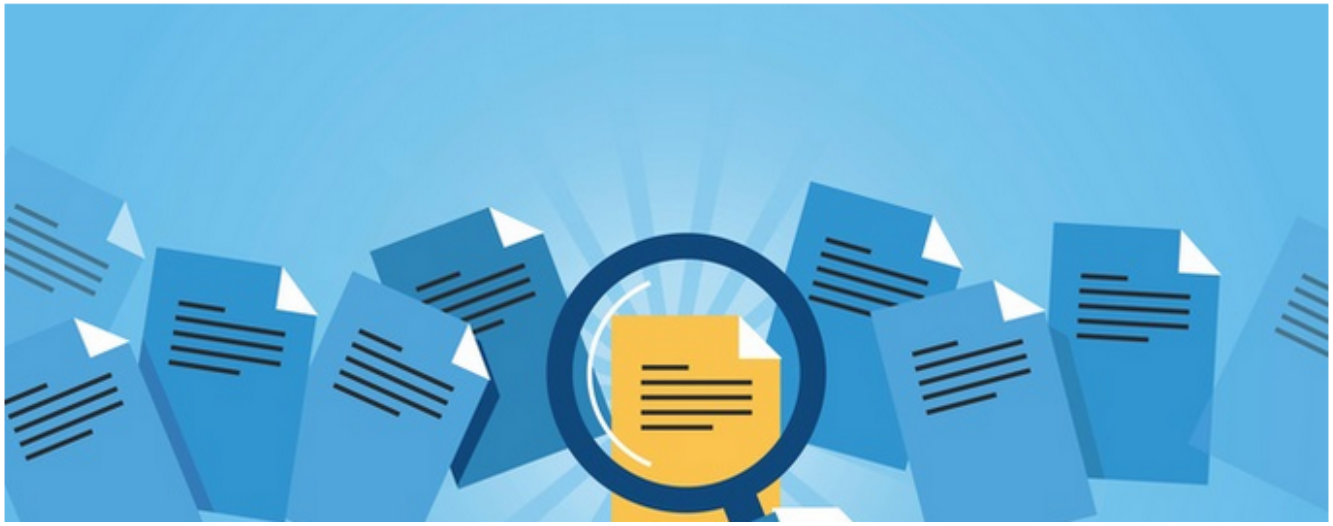
"Match Group's OkCupid and Tinder, for example, were found to be sharing data with each other, including information on their users' sexualities, drug use, and political views... "

Link to Article

...Except When It's Not

Healthcare Big Data Silos Prevent Delivery of Coordinated Care

Healthcare big data silos make it nearly impossible for providers, pharmacies, and other stakeholders to work together for truly coordinated care.



Personal Information

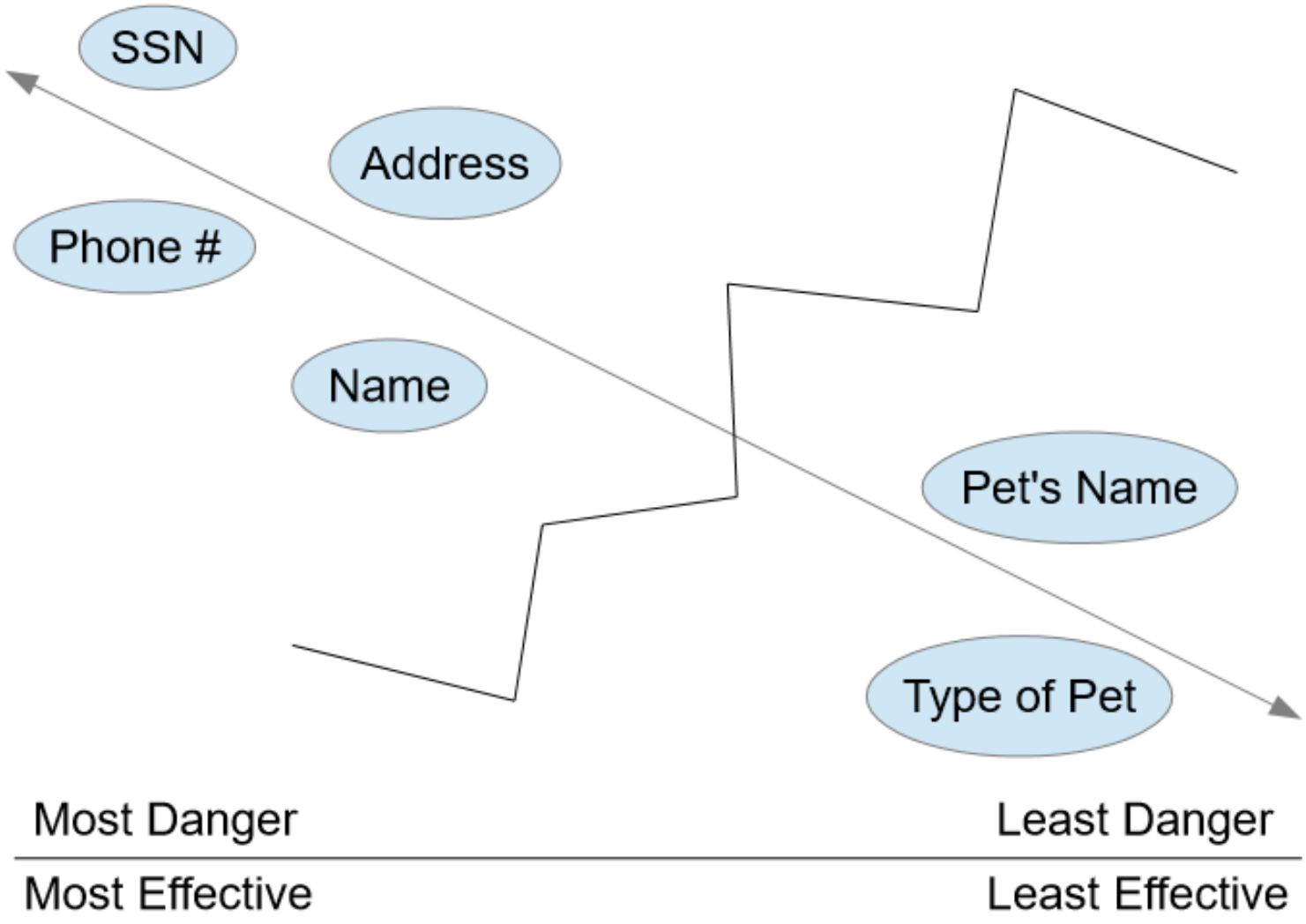
You would immediately recognize things like **name, address, etc.** as private information a business shouldn't lose.

Information that **identifies a specific person apart from another** has a special role in data privacy because identifying an individual is a key part of most crimes you might commit using stolen data.

Many businesses use this personally identifiable information (sometimes abbreviated **P.I.I.**) for these record linkage purposes but **might not actually otherwise need the data**.

You only *need* an address when you print it on an envelope.

Big PII


SSN
Address
Phone #
Name
Pet's Name
Type of Pet
Most Danger
Least Danger
Most Effective
Least Effective

A Familiar Cliche

A phone number can be reliable PII for long periods of time before suddenly becoming worse than useless...

GHOST PII

A Personal Story

One's name can actually be pretty weak PII...

A University of Michigan student accused of sucker-punching a Notre Dame student before the schools squared off on the football field in September was arraigned today.

Alexander G [redacted] **Mueller**, 21, is charged with aggravated assault, Ann Arbor police said. If convicted of the misdemeanor charge, he faces up to a year in jail.

I attended this football game but did not assault anyone. I started giving out my middle initial (which for the record is **C**) because people kept contacting me about the assault anyway.

Link to Article

Record Linkage: Business

Record linkage is finding records in a dataset that refer to the same entity occurring across different data sources.

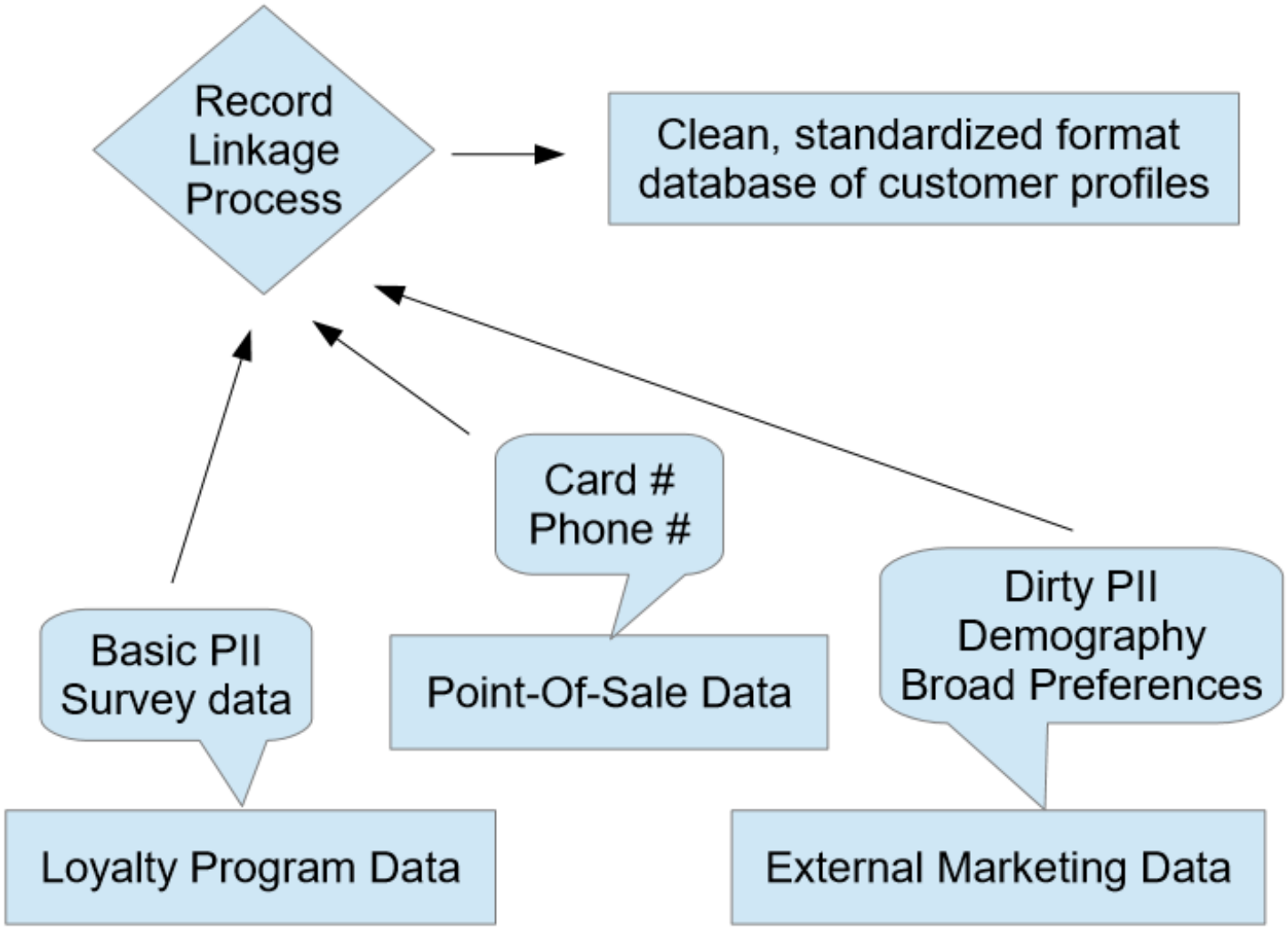
Target had a famous breach. They likely had sources like...

- point-of-sale data
- customer loyalty program data
- external data from aggregators

It might not always be easy to correlate **specific, individual humans** across these three sources.

Data quality is often a stumbling block - even silly things like “Street” in one address source and “St.” in another.

Record Linkage: Data Flow



Record Linkage: Feature Space for ML

Record linkage has a prototypical data wrangling stage.

Start with a data frame on entities (people, businesses, etc.)

- Mostly string data presumed to be a little dirty
- (Many) rows (problem is interesting at scale)

End with a data frame describing pairs of entities

- Columns compare name to name, address to address, etc.
- Clean numeric data describing similarity
- (Many $\times$ Many) rows, depending on blocking

Classify the pairs match, non-match, and maybe-match.

Hospital Reimbursement Raw

A dataframe of records describing hospitals...

	Account_Num	Facility Name	Address	City	State	ZIP Code	County Name	Phone Number	Hospital Type	Hospital Ownership
0	71730	SAGE MEMORIAL HOSPITAL	STATE ROUTE 264 SOUTH 191	GANADO	AZ	86505	APACHE	(928) 755-4541	Critical Access Hospitals	Voluntary non-profit - Private
1	70116	WOODRIDGE BEHAVIORAL CENTER	600 NORTH 7TH STREET	WEST MEMPHIS	AR	72301	CRITTENDEN	(870) 394-4113	Psychiatric	Proprietary
2	87991	DOUGLAS GARDENS HOSPITAL	5200 NE 2ND AVE	MIAMI	FL	33137	MIAMI-DADE	(305) 751-8626	Acute Care Hospitals	Voluntary non-profit - Private

A dataframe of records describing reimbursements...

	Provider_Num	Provider Name	Provider Street Address	Provider City	Provider State	Provider Zip Code	Total Discharges	Average Covered Charges	Average Total Payments	Average Medicare Payments
0	388402	SOUTHEAST ALABAMA MEDICAL CENTER	1108 ROSS CLARK CIRCLE	DOTHAN	AL	36301	118	20855.61	5026.19	4115.52
1	116635	MARSHALL MEDICAL CENTER SOUTH	2505 U S HIGHWAY 431 NORTH	BOAZ	AL	35957	43	13289.09	5413.63	4490.93
2	288613	ELIZA COFFEE MEMORIAL HOSPITAL	205 MARENGO STREET	FLORENCE	AL	35631	73	22261.60	4922.18	4021.79

Feature Space

Below is a simple feature space generated by counting the number of whole-field matches for the city, hospital name, and hospital address fields.

	Account_Num	Provider_Num	City	Hosp_Name	Hosp_Address	Score
0	71395	339408	0	1.0	1.0	2.0
1	26270	868740	1	1.0	1.0	3.0
2	59585	828052	1	1.0	1.0	3.0
3	22555	253626	1	0.0	1.0	2.0
4	22860	560592	1	1.0	1.0	3.0

We might get something more interesting by using more fields and fuzzier matching.

Data Quality and String Metrics

We need a way to formalize our sense that “Meier” and “Maier” might really be the same name.

We might use a **string metric** to assign a numerical score describing how similar one string is to another.

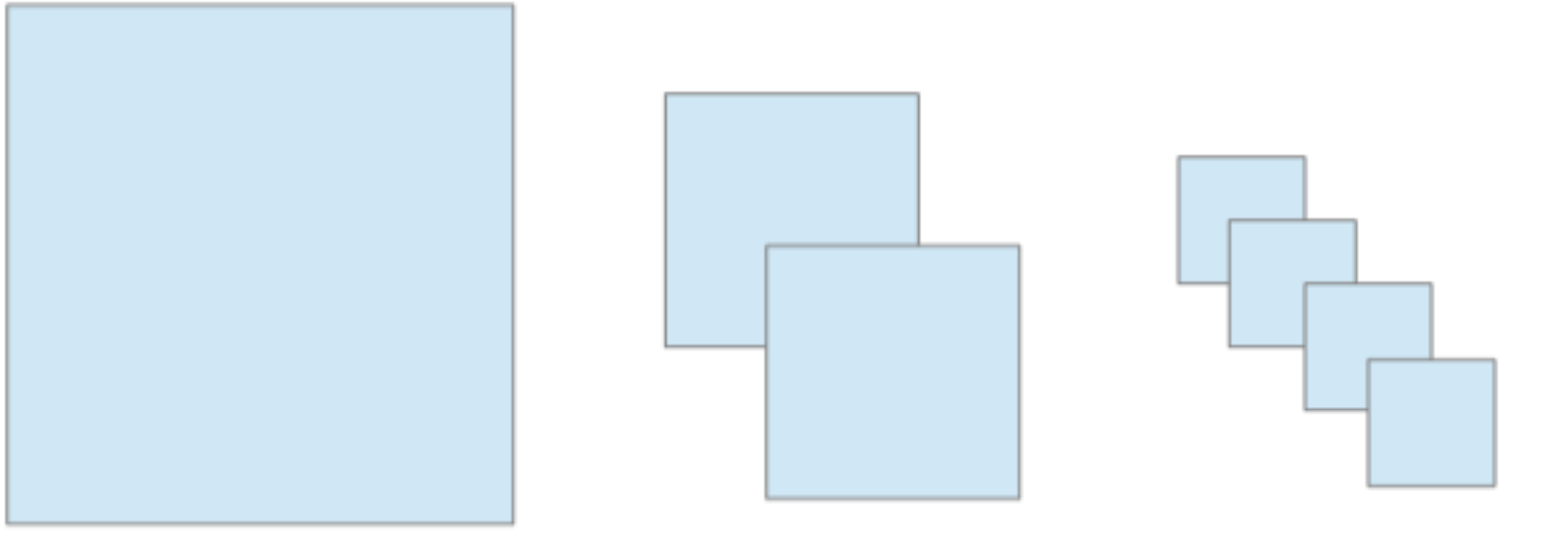
The **Jaro-Winkler distance** is one of many metrics that counts the character edits required to produce one string from another.

There are many other metrics including Levenshtein, bigram, ...

These metrics have meaning even **without knowledge of the underlying strings**.

Blocking: Containing the Feature Space

Record linkage really becomes interesting at scale...



Compare Everyone
N^2 Rows

N/2 at a time
(1/2)*N^2 Rows

N/4 at a time
(1/4)*N^2 Rows

Fellegi-Sunter Theorem

The **Fellegi-Sunter theorem provides some theoretical guarantees** about a **particular probabilistic approach** to classifying pairs as match, non-match, and maybe-match.

Long story short: This approach gives you some weights to compute via an **expectation-maximization (EM)** or Bayesian method.

Many record linkage software packages have this **method built into their interface in some way**, and this can be a little confusing if you don't know to look for it.

There is nothing stopping you from **applying another algorithm** to the feature space of comparison pairs.

Python Tools for Record Linkage

There are a number of relevant tools in the broader numpy and pandas adjacent data ecosystem.

Modules created primarily for record linkage applications

- recordlinkage
- dedupe

Broad-use modules with useful string manipulation tools

- fuzzymatcher
- jellyfish
- nltk

Different organizations might prefer “record linkage” or “entity resolution” to describe the same techniques.

Code From “recordlinkage”

Example code with blocking and string metric choices:

```
compare = recordlinkage.Compare()
compare.exact('City', 'Provider City', label='City')
compare.string('Facility Name',
               'Provider Name',
               threshold=0.85,
               label='Hosp_Name')
compare.string('Address',
               'Provider Street Address',
               method='jarowinkler',
               threshold=0.85,
               label='Hosp_Address')
features = compare.compute(candidates, hospital_accounts,
                           hospital_reimbursement)
```

Dedupe This!

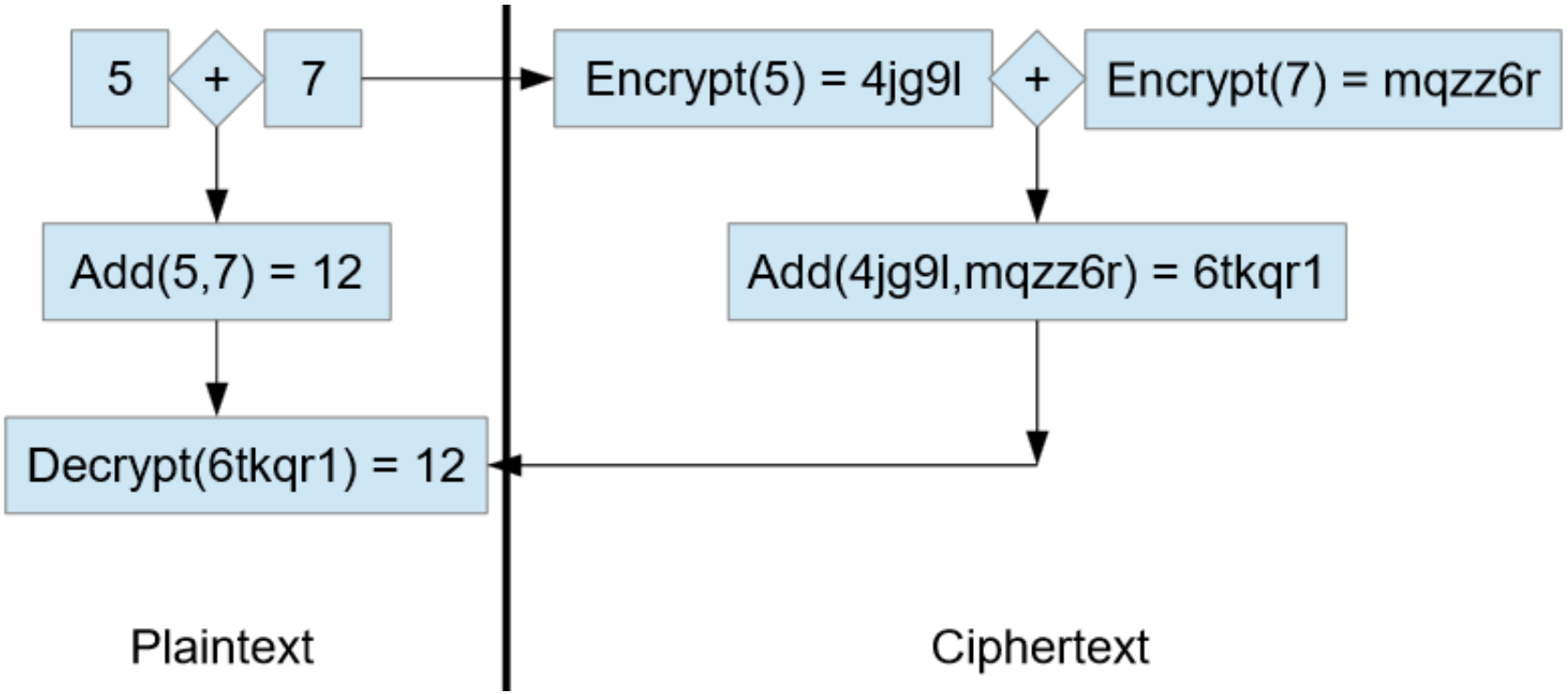
Keeping data encrypted is good for privacy, but how could you do record linkage on data obscured like this?

	0	1
0	#Va%p"#M0"I>"e^!5u	&0""KQ#?E"ZH
1	#fj%Hj%zD"r[%x&%v?	!B@#V8!w9&:,
2	"nG"Cn"F2!pv#9)!]X	#_]%5B!2,&'i
3	%L;#;n%5L"uP!rH!(9	!)p!0_"^@#SE
4	#z8!R8#1u%CE"pF%rg	"l`#ca%^w#1Q

Analytics is about relationships in the data, not the data itself.

GHOST PII

New Tricks: Homomorphic Encryption



Homomorphic encryption allows one to do computations on encrypted data and get the “correct” answer after decryption.

Example: The RL500 dataset...

...from R's RecordLinkage library.

```
> #a random entry
> RLdata500[5,]
  fname_c1 fname_c2 lname_c1 lname_c2   by bm bd uniqueID
5     RALF      <NA>  KRUEGER      <NA> 1966  1 13       72
> #there are lots of muellers out there
> RLdata500[17:19,]
    fname_c1 fname_c2 lname_c1 lname_c2   by bm bd uniqueID
17 ALEXANDER      <NA>  MUELLER      <NA> 1974  9  9       35
18       HANS      <NA> SCHAEFER      <NA> 2003  6 22       88
19    STEFAN      <NA>  MUELLER      <NA> 1949  8 13       77
> #got a dupe because of a type
> RLdata500[RLdata500$uniqueID==444,]
    fname_c1 fname_c2 lname_c1 lname_c2   by bm bd uniqueID
402  CHRISTA      <NA>  SCHWARZ      <NA> 1965  7 13      444
462 CHRISTAH      <NA>  SCHWARZ      <NA> 1965  7 13      444
```

This synthetic data has my name in it. Oy!

Key Idea: Answers Without Data

It would be safer if your data was stored looking like this...

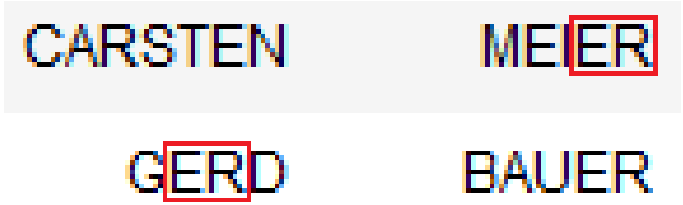
#Cw"H"%Dz%tX#a]"fU!fM%PI#Lf	#6r!L9!3W#8R&6a#7J!3s%rm#eP#vS	!d)#Lb%ra!v2	!h_#Gn	&*e"pI	%PG#ab%#O
"Ip&*c!ti!TU#*f#f:"!+!P9"Zk	!t#&-u!)3%zm"@@"uk#f:"dn">I%'!	"C@!4`!pT"Yn	#W]%ry	"1j&>o	#3'%^7&6G
&!*!>*"R"'_1#;L"qp"e5#aU"??	"!K"Aa%Yr%yO&;0!uR#fM%f5!_@!a!	%Sh"FQ!,e"w9	%NT%WH	"8>"Px	#rw!R*#7u

Does “**as5ga4dsg**” decrypt the same as “**p44hdfj3jdk**”? Y/N?

If we can answer these questions, we can also **compute our string metrics** on encrypted data as they are all based on equality of substrings.

Software often requires these types information but rarely the actual data. It is possible and desirable to provide one (the answer) but not the other (the plaintext data).

Successfully Pretending to be Strings




```
print(myCipherFrame[1][0][3:4]==myCipherFrame[1][0][1:2])
```

```
https://ghostpii.com/recordlink/?lowerOne=7157357&upperOne
other endpoint
True
```

These are objects in Python that...

- contain **data about an encrypted string**
- can be handled with much of the **same syntax** as strings

GHOST PII

Datascience on Top: Compute the Metric

```
wordDistance = myCipherFrame[1][0:150].levenshtein()
```

```
https://ghostpii.com/recordlink/?lowerOne=7338084&upperOn
other endpoint
1.945923089981079
```

```
wordDistance[1]
```

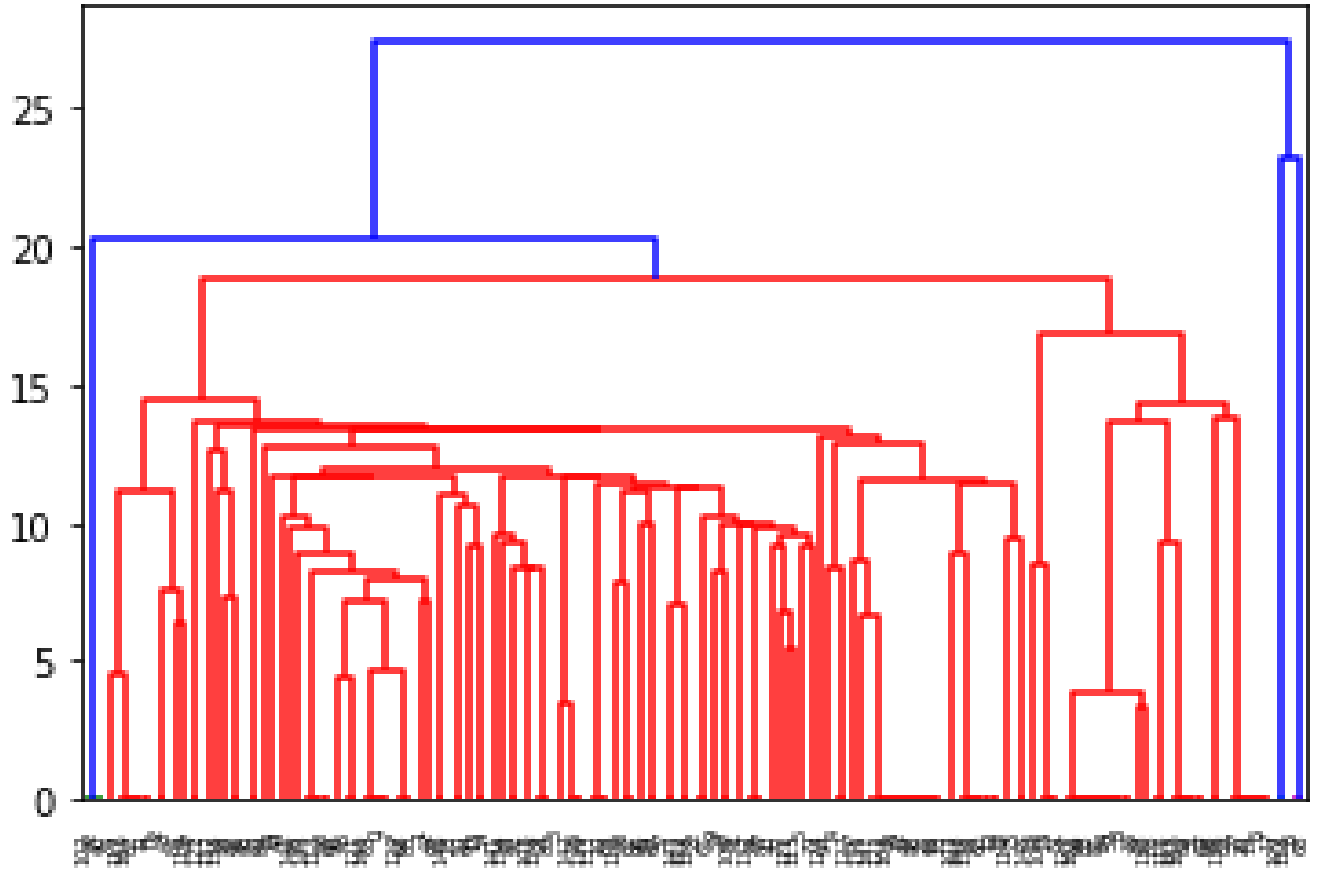
```
array([ 3.,  0.,  7.,  5.,  5.,  6.,  8.,  3.,  7.,  4.,
        4.,  9.,  9.,  6.,  8.,  6.,  8.,  8.,  5.,  5.,
        6.,  8.,  5.,  3.,  7.,  6., 10.,  4.,  4.,  7.,
        7., 10.,  5.,  1.,  7.,  4.,  2.,  6.,  6.,  6.,
        7.,  3.,  7.,  9.,  8.,  6.,  5.,  5.,  7.,  5.,
        3.,  3.,  5.,  9.,  6.,  5.,  8.,  6.,  6.,  4.,
        4.,  5.,  5.,  7.,  6.,  6.,  5.,  3.,  7.,  8.,
```

We are able to compute our string metrics, without seeing the strings, and know how similar each one is to each other.

GHOST PII

Datascience on Top: Visualize Similarity

```
plt.figure()
dn = hierarchy.dendrogram(Z)
```



GHOST PII

Results Computed on Encrypted Data

```
#we'll turn this information into some discrete clusters
from sklearn.cluster import DBSCAN
clustering = DBSCAN(eps=1, min_samples=1, metric='precomputed').fit(wordDistance)
clusterIndex = np.where(clustering.labels_ == 0)[0]
#this is cheating, but will let us see our results
testData = pd.read_csv('rldata500.csv')
testData.iloc[clusterIndex]
```

	fname_c1	lname_c1	by	bm	bd	uniqueID
0	CARSTEN	MEIER	1949	7	22	34
7	UWE	MEIER	1942	9	20	48
45	HERMANN	MAIER	1999	10	12	221
63	JUERGEN	MEIER	1983	7	7	111
66	GERHARD	MEYER	1959	12	3	56
85	GERHARD	MEYER	1941	11	25	30
90	ANDREAS	MAYER	1959	12	24	43

We've used an edit distance and DBSCAN clustering to find groups of pairwise similar surnames.

Questions and Deep Dive

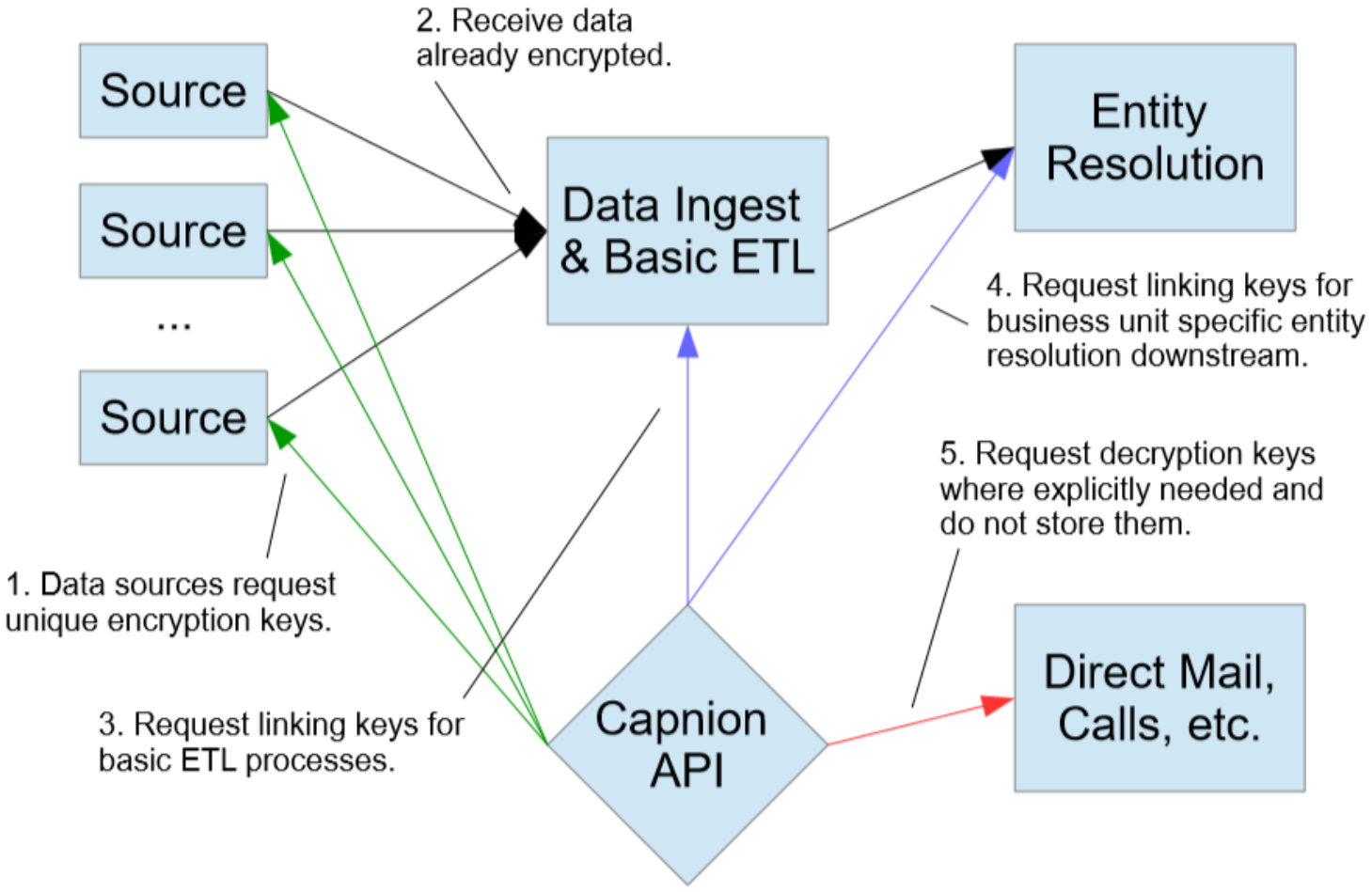
Do you have any questions?

acmueller@capnion.com

Find me on LinkedIn

Slides posted on https://github.com/capnion/random

Appendix: Data Flow and Regulating Insight



GHOST PII

Appendix: About Ghost PII

Everything Capnion does we do **WITHOUT**...

- **seeing or holding your data**, encrypted or not, at any time.
- **requiring you to change your process** for storing and transmitting data.

Capnion's unique Answer Key functionality...

- allows **detailed access control**, by user or row / column
- naturally generates a **data use audit trail**

The idea is to work with encrypted data locally, compute an encrypted answer using that data, and download a special key that **decrypts the result, and only this result**, at the very end.

Appendix: Differentiators vs. Hashing Approaches

Hashing techniques have been used for similar problems, but they have a number of significant drawbacks.

Hashing is typically not secure **without the use of salting**, and governance of salting adds a layer of complexity.

Inadequate salting or governance thereof has in many cases produced **insecure protocols that fell quickly** to attackers.

Less-than-perfect data quality renders hash-based approaches **unreliable or useless**, as the tiniest change in spelling or abbreviation (extremely common in address data, for example) produces entirely distinct hashes. Hash based approaches by their nature **exclude any possibility of fuzzy matching or similarity comparison**.