



**COMPUTATIONAL
PRIVACY
GROUP**

Give us feedback:

This is a new course we are designing, feedback on the content, slides, delivery (too slow, too fast,...) is more than welcome!
florimond@imperial.ac.uk

Privacy Engineering

Week 2 - Big Data anonymization

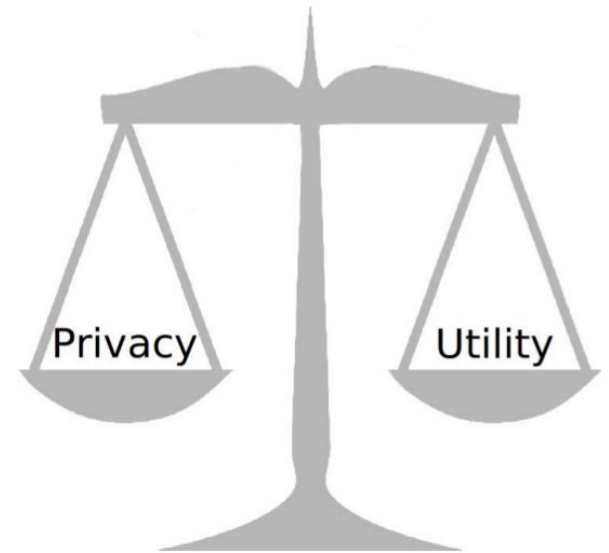
Anonymization is hard

Once the dataset is properly pseudonymized, we need to:

- 1) protect it against a whole range of attacks: uniqueness, homogeneity, semantic, skewness
- 2) by anonymizing it once and only once
- 3) all while preserving utility (all current and future uses).

It's already not simple to make it work for small tabular datasets.

Does it get easier with **Big Data**?

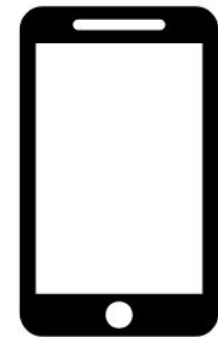
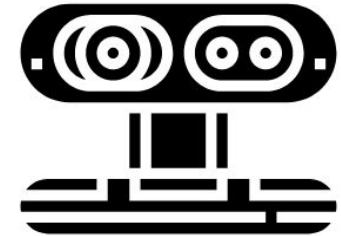


What is Big Data?

Formally (according to Google): “**extremely large data sets** that may be analysed computationally to reveal patterns, trends, and associations, especially relating to **human behaviour and interactions.**”

In short the data is:

- Large and often high-dimensional (“lots of columns”), it *does not fit in Excel*.
- automatically collected as a side-effect of our technology.



H6ycJQIv.csv:

call,in,sW4aFX,2014-03-02 07:13:30,210,42.366944,-71.083611

call,out,5f0jX5G,2014-03-02 07:53:30,34,42.366944,-71.083611

text,in,5f0jX5G,2014-03-02 08:22:30,,42.386722,-71.138778

AnonID	Query	QueryTime	ItemRank	ClickURL
142	rentdirect.com	2006-03-01 07:17:12		
142	westchester.gov	2006-03-20 03:55:57	1	
	http://www.westchestergov.com			
1326	budget truck rental	2006-03-24 18:27:07		
1326	holiday mansion houseboat	2006-03-29 17:14:01	5	
	http://www.everyboat.com			
1326	back to the future	2006-04-01 17:59:28		

Urban Outfitters 7abc1a23 09/23 \$97.30

Market Basket 7abc1a23 09/23 \$15.13

Whole Food 3092fc10 09/23 \$43.78

Central Bakkery 7abc1a23 09/23 \$4.33

MIT RecSport 4c7af72a 09/23 \$12.29

The standard terminology, assumptions, and metrics do not work anymore

The data corresponds to a *behavioural trace* of an individual:

- There is no *sensitive attribute*: every point in the data is potentially sensitive.
- There is no *quasi-identifier*: no point or combination of points that clearly identifies every individual (and can be assumed to be known by an adversary).

For instance, take **location data**:

- Every point is potentially sensitive: you don't want anyone to learn your real-time location.
- No point is special, in that it can be assumed to be known or to uniquely identify you.

Hence, the **standard definitions** of small scale data anonymization **does not work anymore**. We need new definitions and metrics.



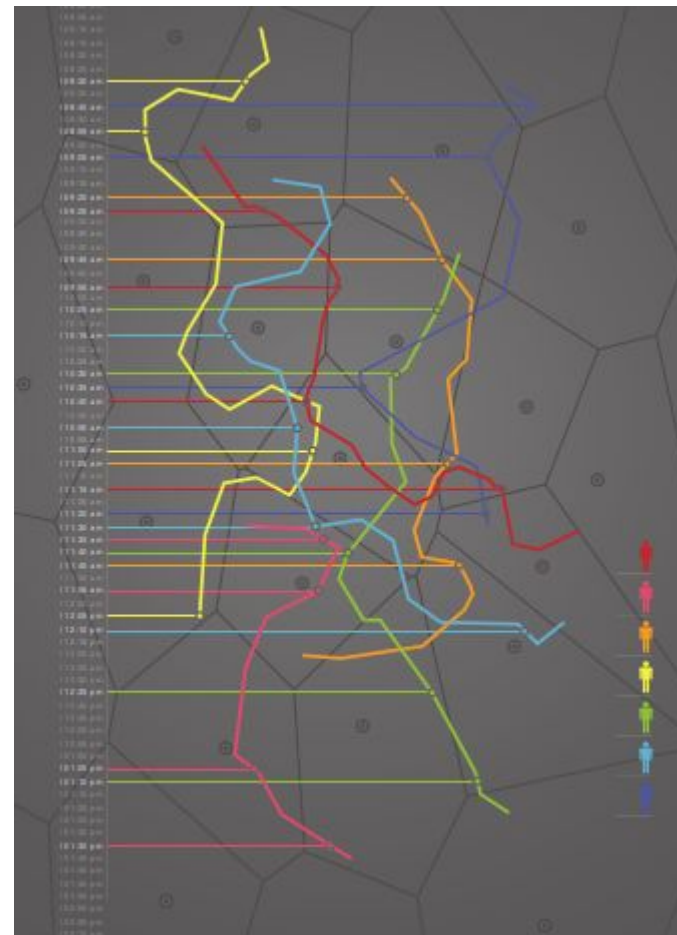
From uniqueness to unicity ε_p

No quasi-identifiers or sensitive data anymore, every point is both sensitive and a point that could be known to an attacker.

However, we don't assume an attacker knows all the points, just a few (p) of them

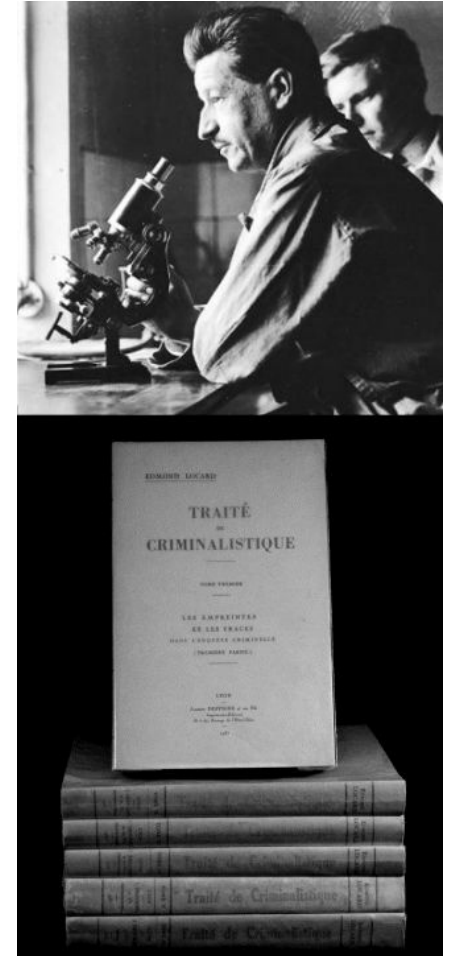
Unicity (de Montjoye, 2013) aims at quantify the **risk of re-identification** in large scale behavioural datasets.

Formally, the **unicity ε_p of a dataset** is the average fraction of the users in the dataset that are uniquely identified by p random points in their trajectory



The notion of point comes from fingerprinting

In 1930, Edmond Loccard showed that 12 points are necessary to **uniquely** identify a fingerprint

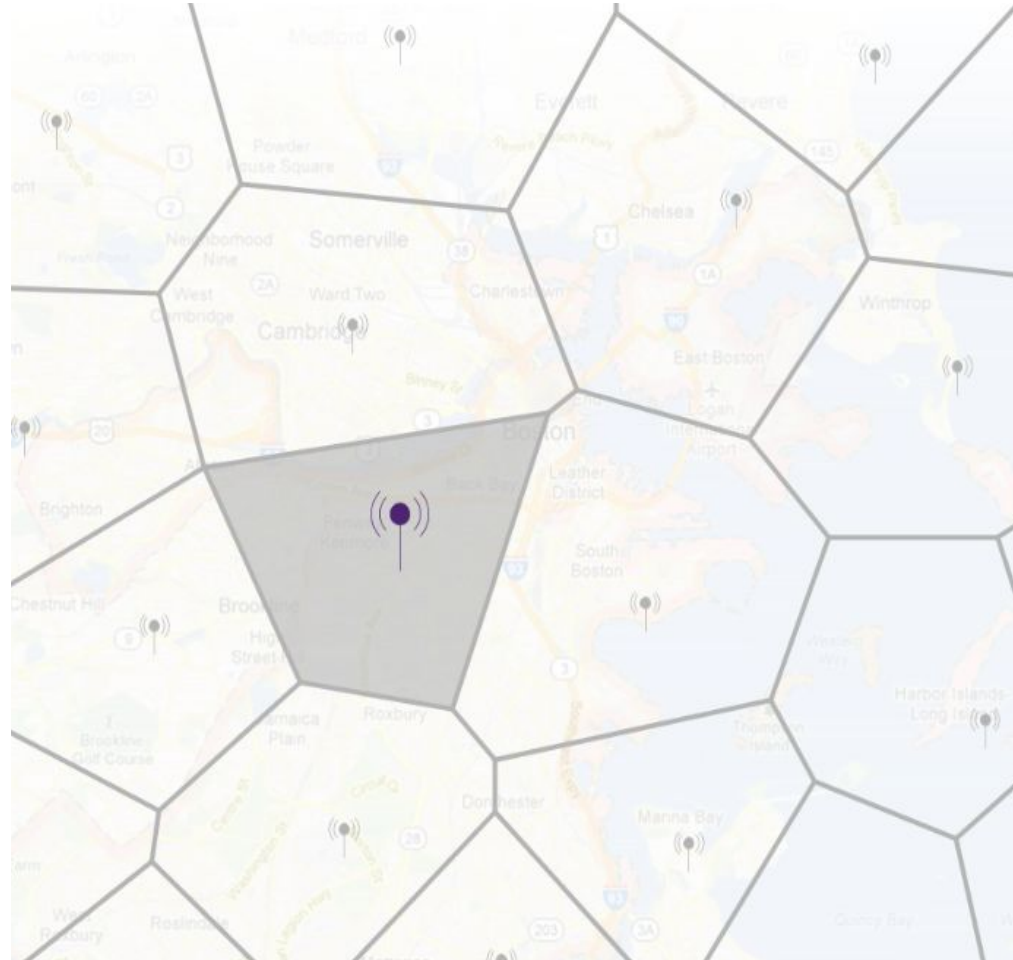


A point in location data

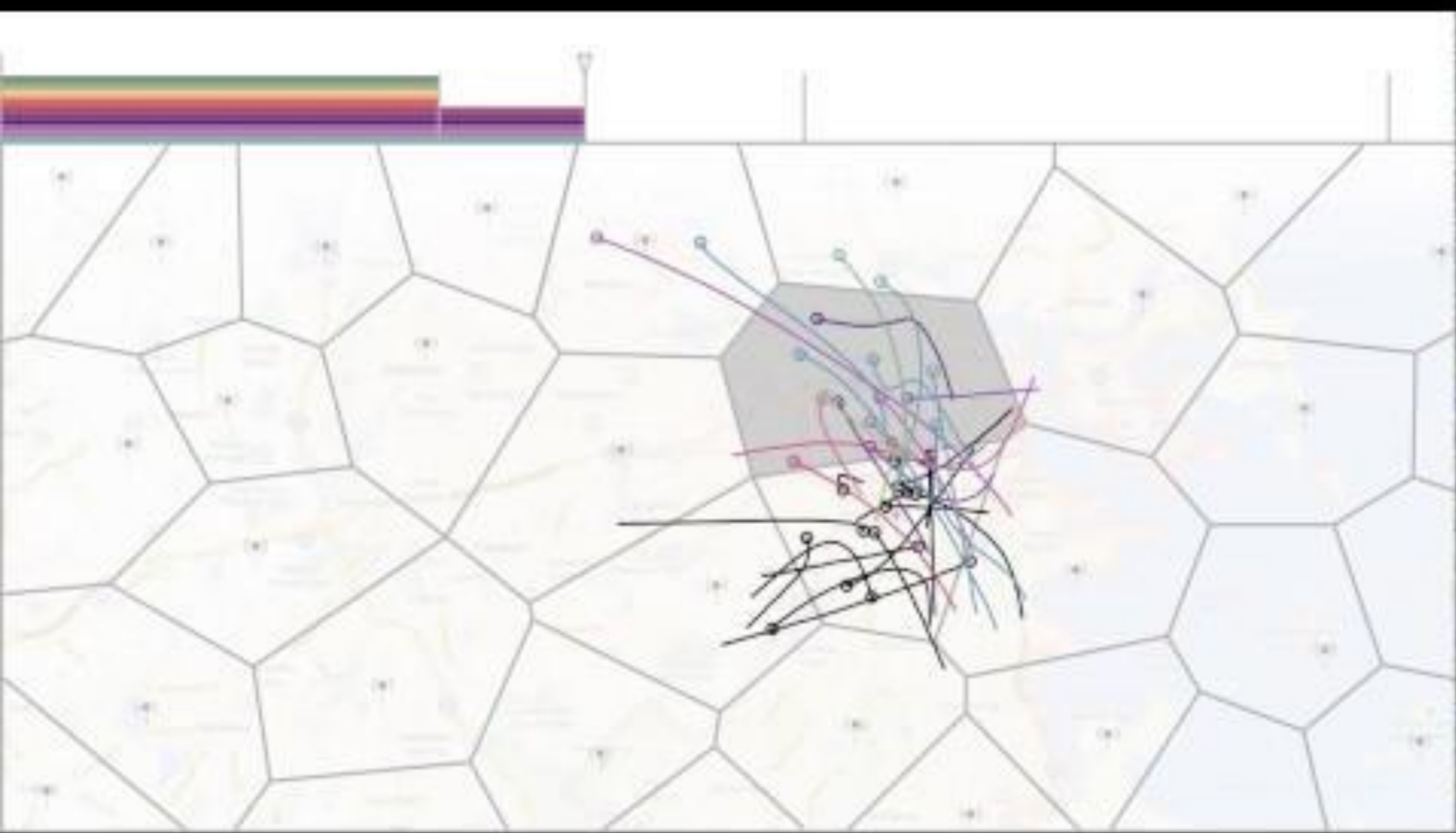
In the unicity attack model, the attacker's auxiliary information are points.

In location data, a point would be a pair *(location, time)*, i.e. a place (more precisely a region like the one covered by an antenna) and a time (e.g. between 10 and 11am).

We assume (as before) that the target is in the dataset and that the auxiliary information are correct



How many points do we need?



Unicity of mobile phone metadata

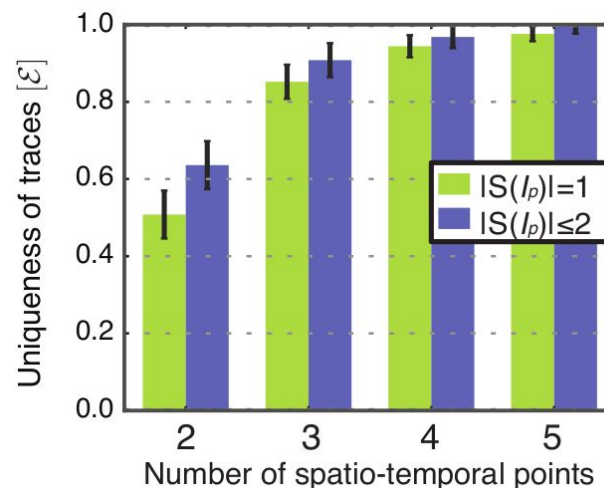
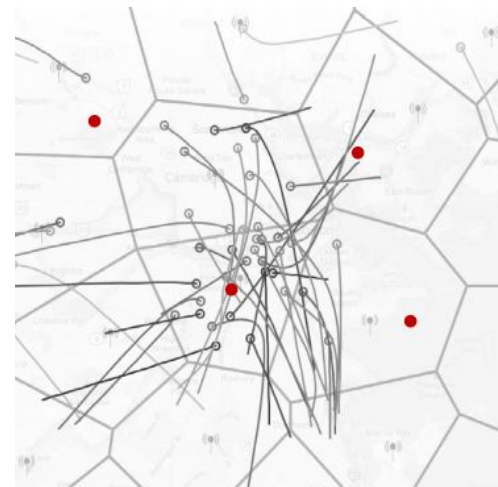
In a dataset of 1.5M people over 18 months, we found that:

$$\epsilon_4 = 0.95$$

In other words: 95% of people are unique for 4 points of their trace.

Even with two points we identify more than 50% of the people

Finally, we found that, out of 1.5M, 13 points were necessary to identify everyone in a sample of 10.000 people

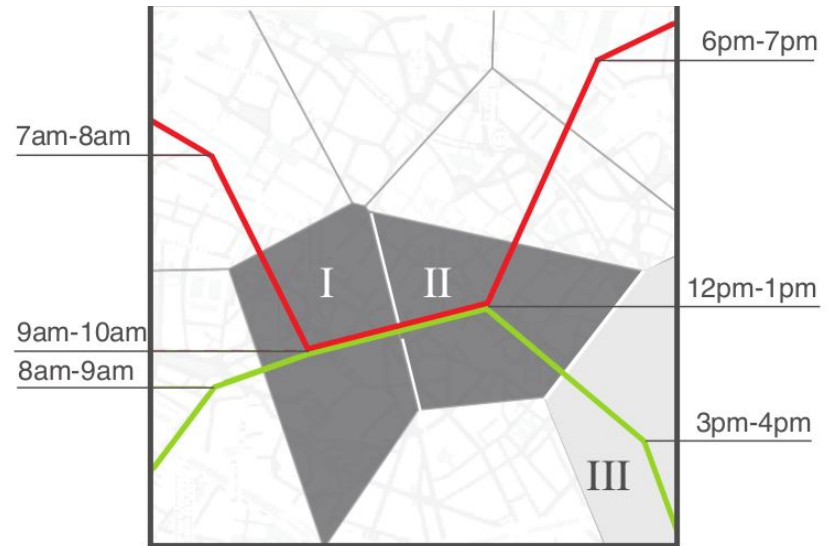


Estimating unicity

Computing unicity exactly would take a very very long time, one would need to compute for all users ($n = 1.5M$) and for all set of points from their traces (~ 1000 choose p) whether this makes them unique or not.

Instead, we used the following procedure that relies on:

- A random set of 10000 users
- For which we draw p points at random
- And whether these points make them unique



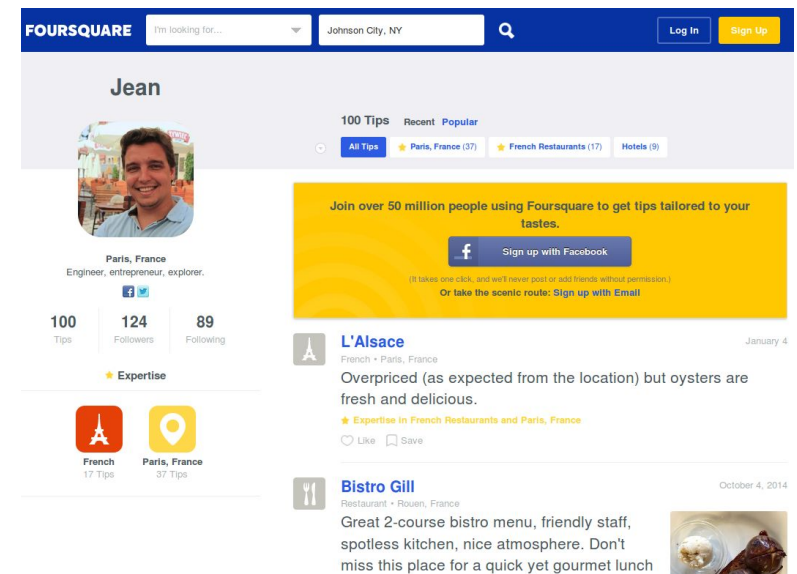
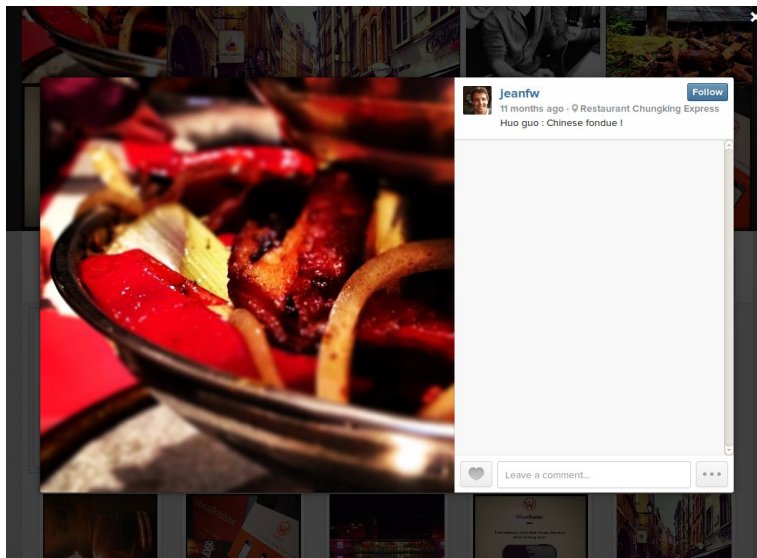
Algorithm Unicity Estimation(p)

```
1:  $users \leftarrow \text{select}(D, p, 10000)$ 
2: for  $u \in users$  do
3:    $I_p \leftarrow \text{draw}(u, p)$ 
4:    $is\_unique \leftarrow \text{true}$ 
5:   for  $x \in D \setminus \{u\}$  do
6:     if  $I_p \subset x.trace$  then
7:        $is\_unique \leftarrow \text{false}$ 
8:       break
9:   if  $is\_unique$  then
10:     $uniqueUsers \leftarrow uniqueUsers + \{u\}$ 
11: return  $|uniqueUsers| / |users|$ 
```

Where could an attacker collect points
(place and time where you were from)?

Offline but also online

We constantly broadcast our location on many services (Twitter, Facebook, Foursquare, Instagram, ...)



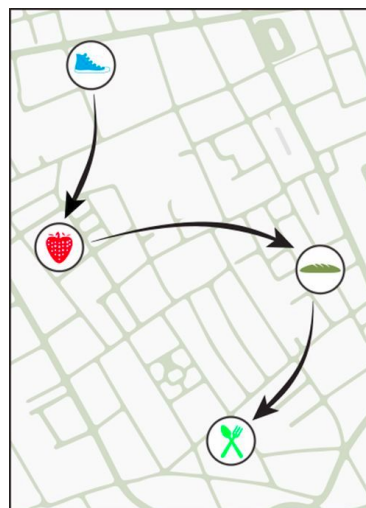
Unicity of credit card data

Points are here a shop (place where you paid using your credit card) and a day

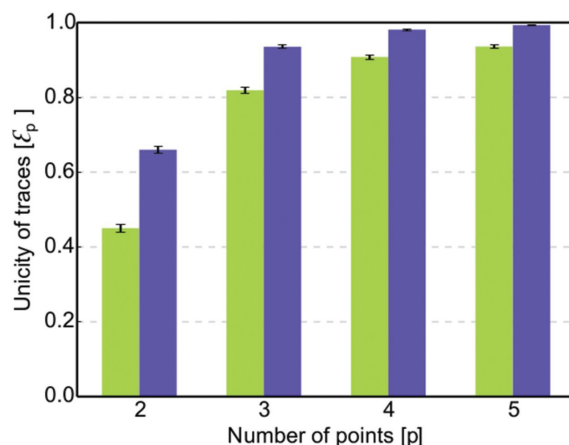
In a dataset of 1.1M people over 3 months, we found that:

$$\epsilon_4 = 0.90$$

Using additional information about the price of the transaction further increase unicity by 22% on average



shop	user_id	time	price	price_bin
	7abc1a23	09/23	\$97.30	\$49 – \$146
	7abc1a23	09/23	\$15.13	\$5 – \$16
	3092fc10	09/23	\$43.78	\$16 – \$49
	7abc1a23	09/23	\$4.33	\$2 – \$5
	4c7af72a	09/23	\$12.29	\$5 – \$16
	89c0829c	09/24	\$3.66	\$2 – \$5
	7abc1a23	09/24	\$35.81	\$16 – \$49



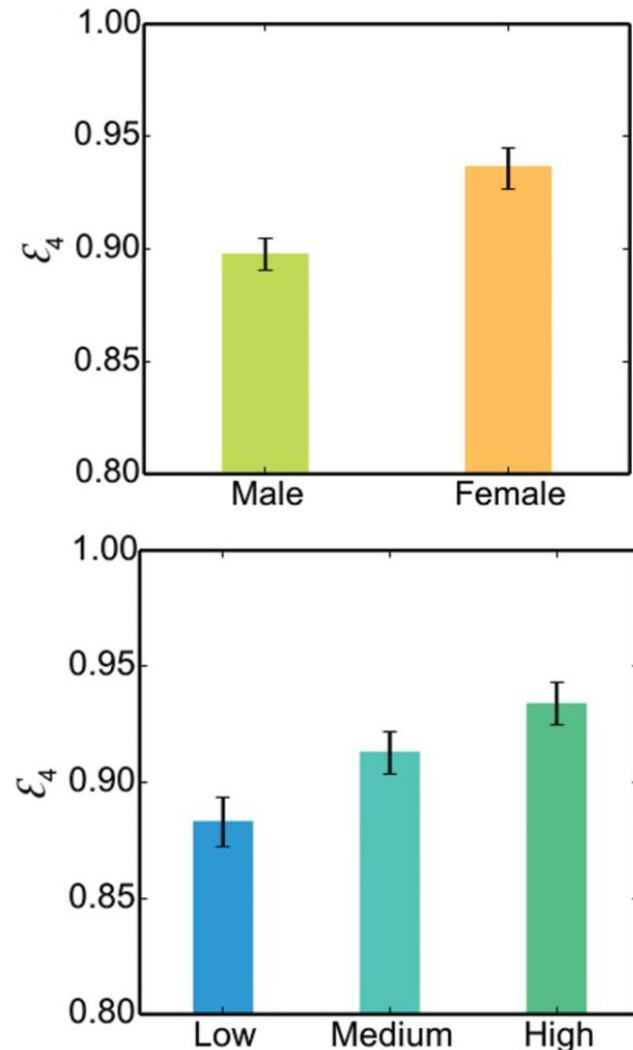
Green bars are unicity when knowing the day and place of the transaction. Blue bars when one also knows the approx. amount of the transaction

What does unicity tell us about behavior

Unicity measures what distinguishes one person from literally everybody else in the dataset.

Looking at unicity in credit card data we showed that

1. Women are easier to identify than men. (Odds of women to be unique is 1.214 times greater than for men)
2. The richer you are, the easier you are to identify. (Odds of high income people to be unique 1.746 times greater than for low income people)



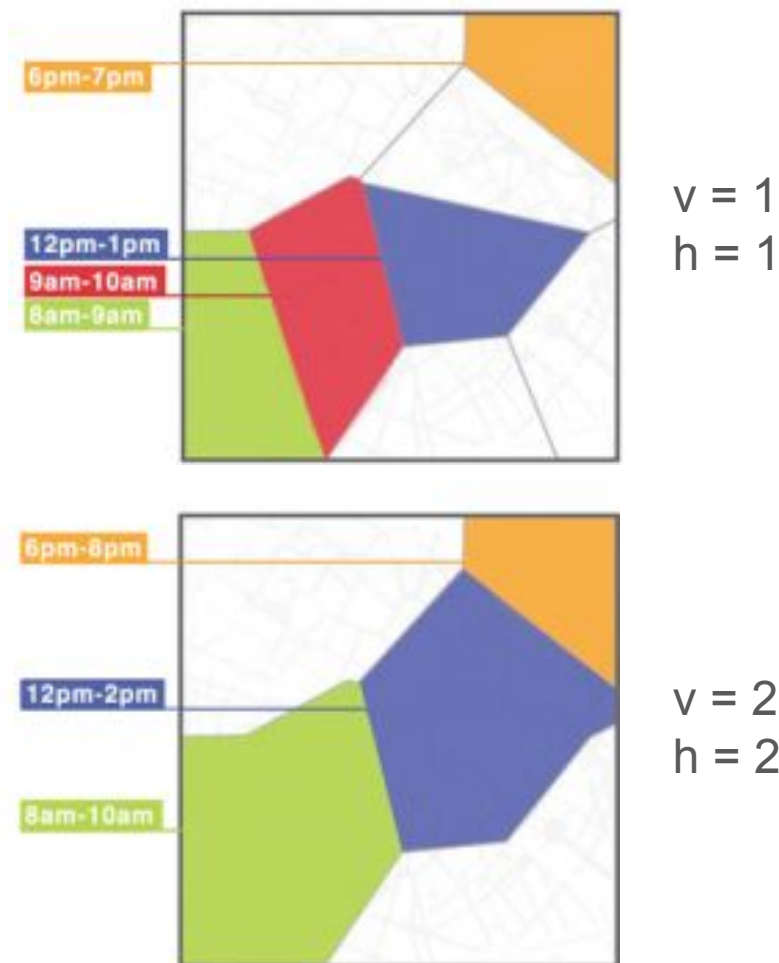
Can we solve this with generalization?

We know that individuals are **highly identifiable** in large-scale datasets.

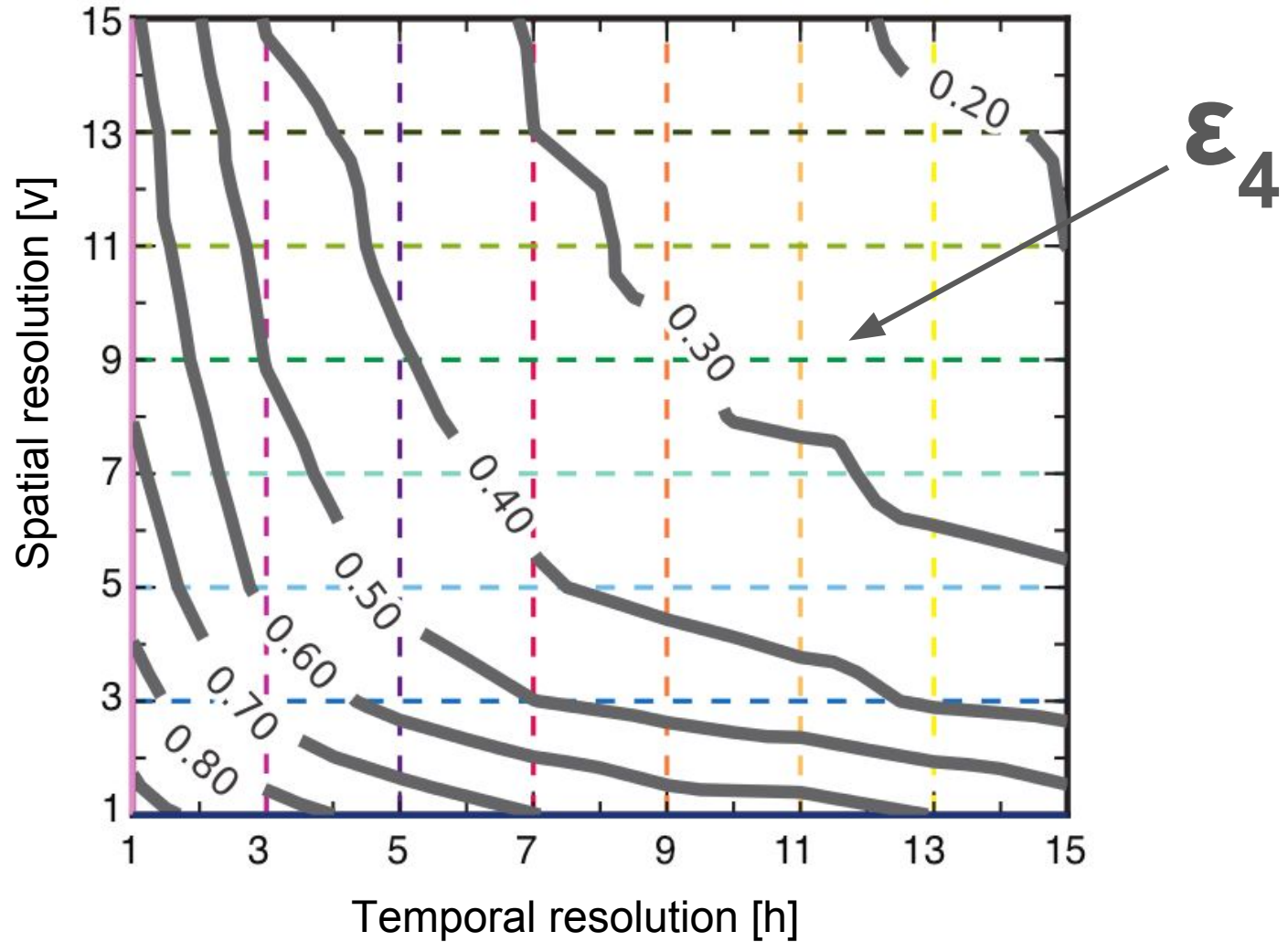
Generalization helped achieve k-anonymity in small dataset → can we use it for Big Data?

Idea: coarsen the data by reducing spatial and temporal resolution. Instead of reporting the precise antenna, give a more general area; instead of hours, give time periods.

Is this going to be sufficient?
What would you consider sufficient?



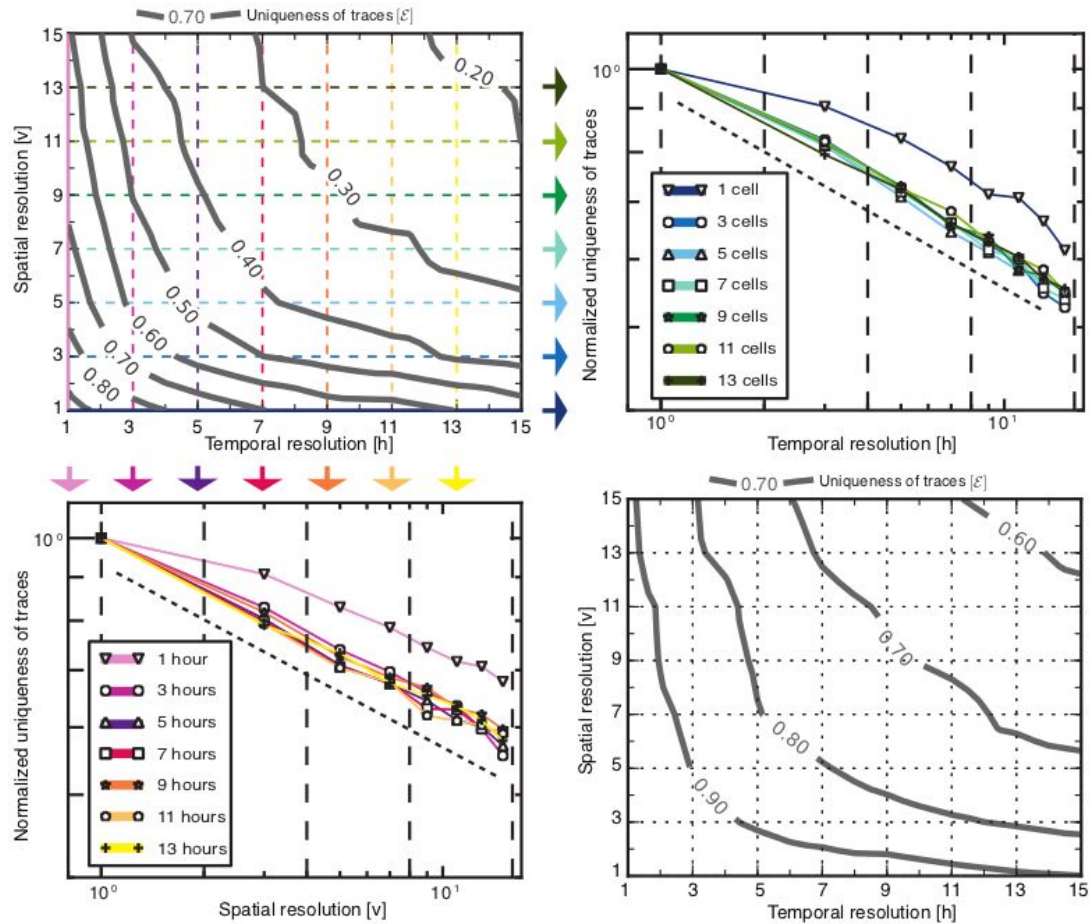
Generalisation does help



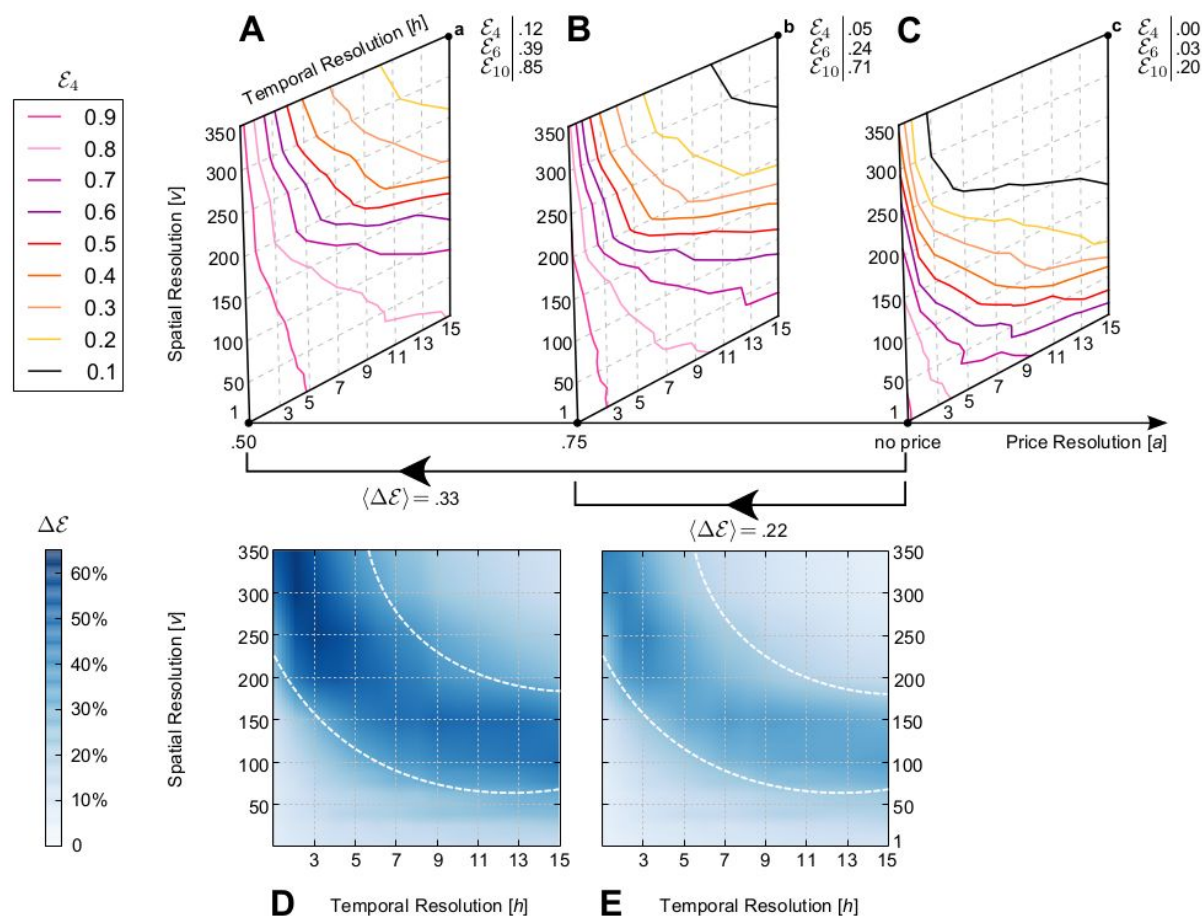
But it is not sufficient for mobile phone data

- 1) We can see that we have “decreasing return” on privacy when decreasing the spatial and temporal resolution of the data
- 2) Even when the data is very coarse, only a few more points are enough to uniquely identify a large number of people

$$\mathcal{E} \sim (v * h)^{-p/100}$$



Nor for credit card data



de Montjoye Y.-A., et al., Unique in the shopping mall: On the reidentifiability of credit card metadata. Science 347 (6221), 536-539. (2015).

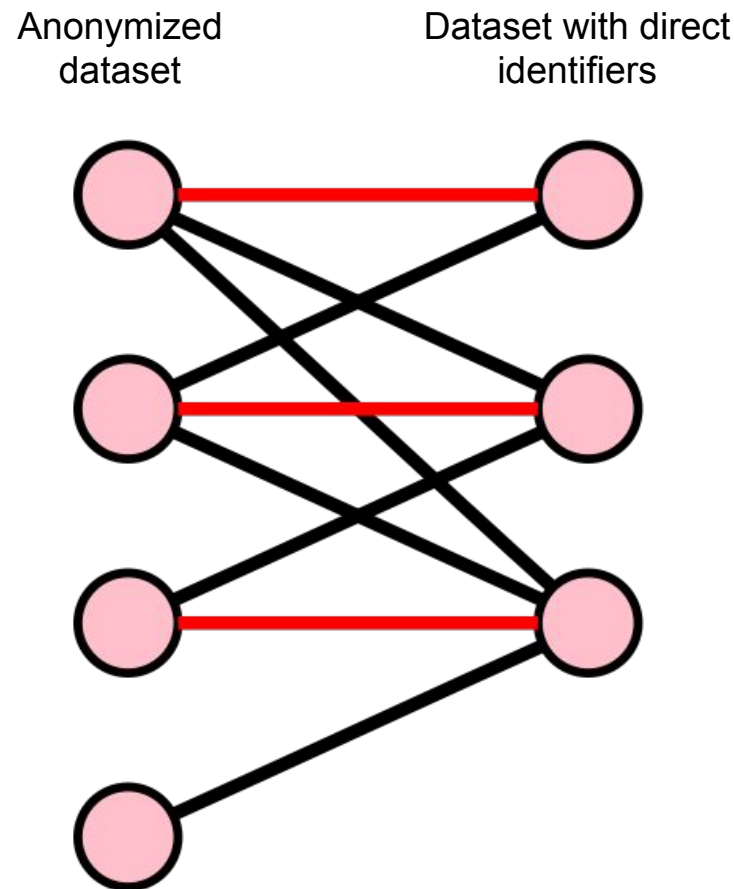
But there might be noise:
matching attacks

Matching attacks

Auxiliary information might not directly match the information available in the anonymous dataset, the data could be noisy, missing or match several people. Similarly, the person we're searching for might not be in the dataset.

Matching attacks are implementations of unicity that rely on:

1. A measure of **distance**, measuring how similar two records are;
2. A **linking algorithm** to perform the decision, based on the metric.

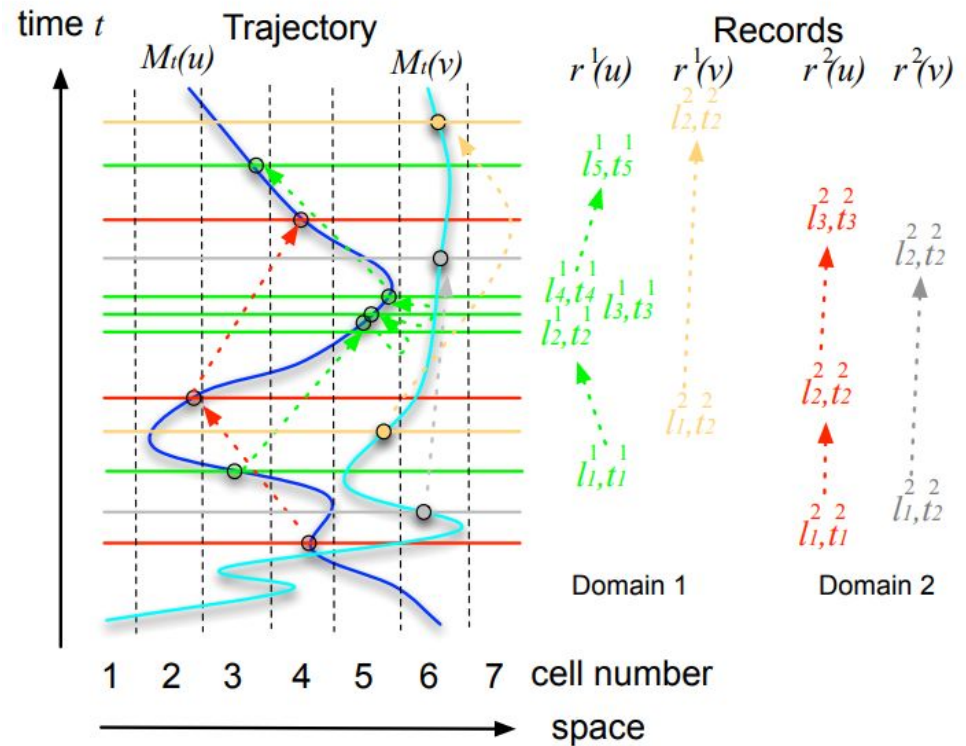


Matching attack: location data

Why going beyond simple matching (matching to the trace with the most points in common)?

Model: while people move, they perform “actions” that are recorded in one of the datasets (they send a text message, use an app, ...).

Assumption: the number of actions $A(u, l, t)$ a user u performs in a region l and a time interval t is distributed according to a Poisson distribution of parameter $\lambda_{(l,t)}$



Matching attack: location data

Step 1: for each user u in the identified dataset and each user v in the anonymised dataset, compute a **score** describing how close u and v are.

This score is computed using probabilities (under the Poisson assumption), as:

$$w(u, v) = \sum_l \sum_t (\log \varphi_{l,t}(u,v))$$

Where $\varphi_{l,t}(u,v)$ is the ratio of probabilities of (1) if u and v are the same user and (2) if they are independent:

$$\frac{P[A_1(u, \ell, t) = a_1 \wedge A_2(v, \ell, t) = a_2 \mid \sigma_I(u) = v]}{P[A_1(u, \ell, t) = a_1] \cdot P[A_2(v, \ell, t) = a_2]}$$

Matching attack: location data

Probability that user u has a_1 actions AND user v has a_2 actions **together**...

.. if they are **the same user**

$$\frac{P[A_1(u, \ell, t) = a_1 \wedge A_2(v, \ell, t) = a_2 \mid \sigma_I(u) = v]}{P[A_1(u, \ell, t) = a_1] \cdot P[A_2(v, \ell, t) = a_2]}$$

permutation

Probability that user u has a_1 actions (alone).

Probability that user v has a_2 actions (alone).

This ratio is **high** if the fact that u and v are the same user makes (a_1, a_2) more likely than if they were independent users.

For instance, if a_1 and a_2 are particularly large, if u and v are the same user, then the numerator becomes high, where the denominator is small.

Nice property:

Thm: the log-likelihood of a matching σ is proportional to $\sum_{u, v=\sigma(u)} \mathbf{w}(u, v)$

Matching attack: location data

Step 2: Compute the **maximum weight matching** between people in U and people in V .

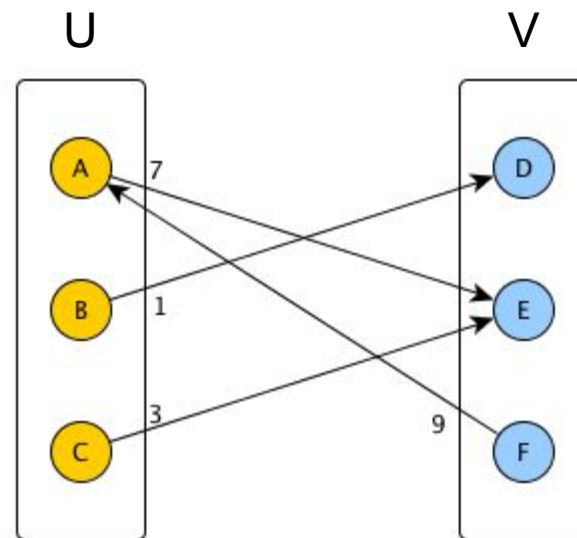
Complexity (to create the graph): $O(|U||V|)$

This is done using the **Hungarian algorithm**.

Complexity: $O((|U|+|V|)^3)$

Thm: under the Poisson assumption, the true matching between U and V has maximum expected likelihood (among all matchings)

Step 3: An edge is considered a match only if its score differs from the second-best score by more than ϵ times (eccentricity) the standard deviation of all other possible matches' score (for this user)



If there were $u = |U|$ people in U and $v = |V|$ people in V with $u \neq v$ how many matches are we going to have?
A) $u+v/2$; B) $\min(u,v)$; C) $\max(u,v)$

Evaluating it on real data (1)

1. The datasets

Dataset	Domain	Users	Checkins	Median Checkins	Locations
FSQ-TWT	Foursquare	862	13,177	8	11,265
	Twitter	862	174,618	60.5	75,005
IG-TWT	Instagram	1717	337,934	93	177,430
	Twitter	1717	447,366	89	182,409
Call-Bank	Phone Calls	452	~200k	~550	~3500
	Card Transactions	452	~40k	~60	~3500

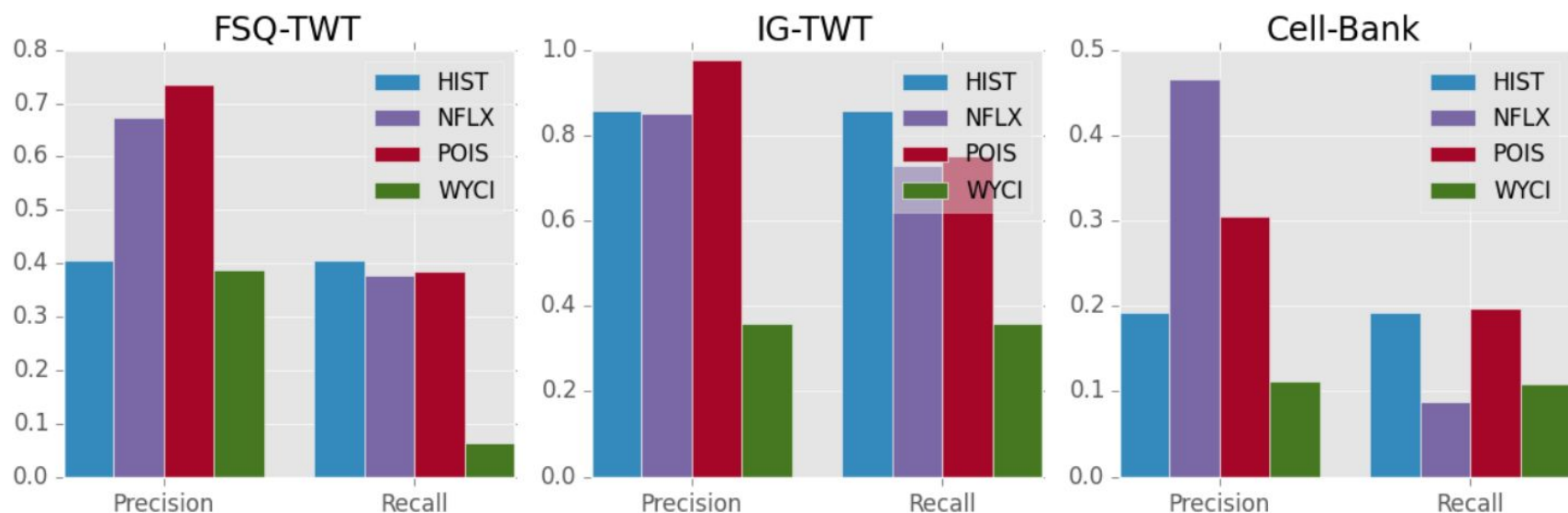
2. The metrics

→ Let N_{match} the number of matches returned by the algorithm, N_{true} the number of true matches initially in the data and $N_{\text{true+match}}$ the number of true matches found.

- Precision: $N_{\text{true+match}} / N_{\text{match}}$ (“*how likely is it that a match is true*”)
- Recall: $N_{\text{true+match}} / N_{\text{true}}$ (“*how many true matches did we find?*”)

Evaluating it on real data (2)

They compare it to three other algorithms that exist in the literature (we'll cover two of them later on): an histogram matching alg. (HIST), the algorithm used against the netflix dataset (NFLX), and a frequency-based one (WYCI)



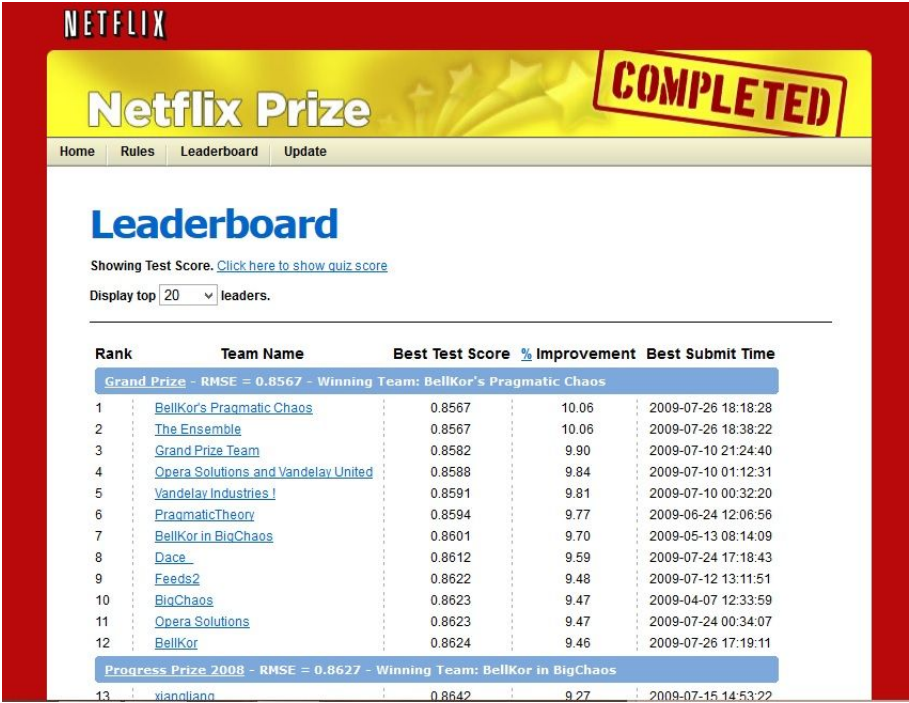
How is the algorithm performing?

Why could the algorithm perform worse on the last example (Cell-Bank)?

Matching attack on movie reviews

In 2006, Netflix published an anonymous dataset of movie reviews left by 500.000 of its users. The goal was for researchers to develop a better recommendation algorithm (for a prize of \$1M).

Narayanan and Shmatikov re-identified some of the users using a **matching attack** with users' (public) IMDB profile as auxiliary information (name of the movie, time of watching and review)



The screenshot shows the Netflix Prize Leaderboard interface. At the top, the Netflix logo is on the left, and a yellow banner with 'Netfix Prize' and a 'COMPLETED' stamp is on the right. Below the banner are navigation links: Home, Rules, Leaderboard, and Update. The main heading is 'Leaderboard'. Below it, it says 'Showing Test Score. [Click here to show quiz score](#)'. There is a dropdown menu for 'Display top' set to '20' and a link to 'leaders.'. The table below lists the top 12 teams for the Grand Prize. The winning team is BellKor's Pragmatic Chaos with a RMSE of 0.8567. The table also shows the improvement percentage and the best submit time for each team.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries I	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xianqiang	0.8642	9.27	2009-07-15 14:53:22

Narayanan, A., & Shmatikov, V. (2008, May). Robust de-anonymization of large sparse datasets. In Security and Privacy, 2008. SP 2008. IEEE Symposium on (pp. 111-125). IEEE.

Matching attacks: the Netflix dataset

Distance measure: (or in this case, similarity between auxiliary info Aux and a record r in the dataset)

$$score(Aux, r) = \sum_{i \in supp(Aux)} \frac{1}{\log |supp(i)|} Sim(Aux_i, r_i)$$

With:

- $Sim(x_i, y_i) = 1$ if the movie i is in both x and y , otherwise 0.
 - $supp(i) =$ set of all users that watched the movie i
 - $supp(Aux) =$ set of movies in Aux
- Why could dividing by $supp(i)$ be useful here?

Linking algorithm: With r^* the record of higher score and r_2 the record with second best score. We consider r^* to be a **match** if:

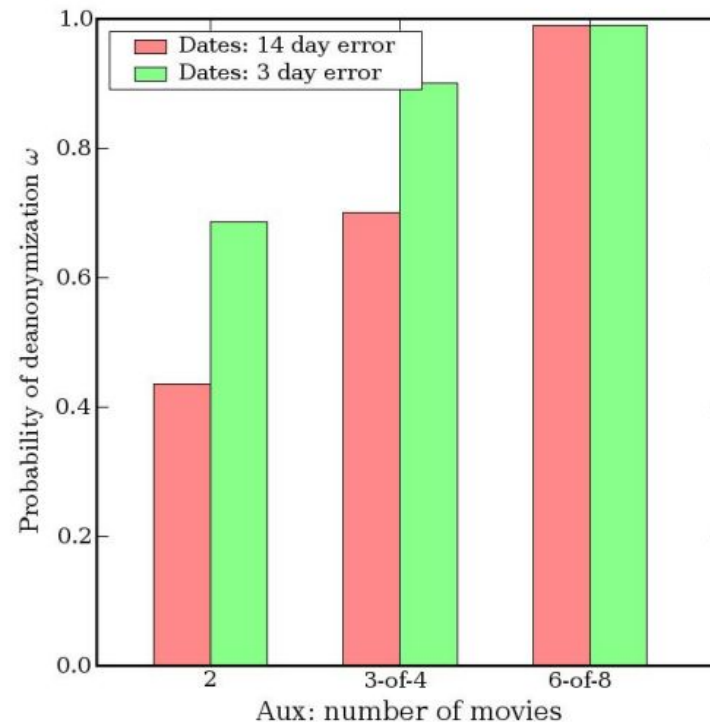
$$Score(aux, r^*) - Score(aux, r_2) > \varphi \text{ var(scores)}$$

with φ a parameter called the **eccentricity**, and $scores$ is the list of scores obtained when comparing with the dataset: $[Score(aux, r): r \in dataset]$.

Evaluating it on the Netflix dataset

They first used the matching algorithm on the dataset assuming they knew the **exact rating** but only the **approximate date the movie had been watched**. Most users would be re-identified if the attacker were to know 6 movies they watched.

Second, they matched the data from a few dozen IMDB profile and were able to identify two users with (very) high confidence (using high eccentricity, they don't have ground truth).



The popularity of movies is heavily tailed (few popular movie and a lots of niche one), if popularity were to be uniformly distributed would that make it easier or harder to re-identify someone? A) easier; B) no effect; C) harder

But what if the data doesn't overlap timewise?

Dataset	Domain	Users	Checkins	Median Checkins	Locations	Date Range
FSQ-TWT	Foursquare	862	13,177	8	11,265	2006-10 – 2012-11
	Twitter	862	174,618	60.5	75,005	2008-10 – 2012-11
IG-TWT	Instagram	1717	337,934	93	177,430	2010-10 – 2013-09
	Twitter	1717	447,366	89	182,409	2010-09 – 2015-04
Call-Bank	Phone Calls	452	~200k	~550	~3500	2013-04 – 2013-07
	Card Transactions	452	~40k	~60	~3500	2013-04 – 2013-07

Matching attacks like the two we just looked at won't work anymore

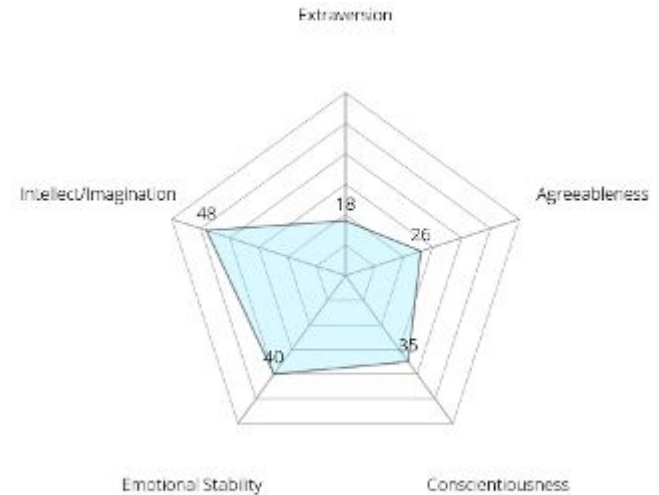
We need what are called **profiling attacks**, where the distance is time independent. Profiling methods typically rely on user's habits and behavior to re-identify him or her.

Profiling attacks

Problem: Identifying users in an *anonymous* dataset using an *identified dataset* collected at a different time.

Methodology:

1. Extract a profile of the user in the *identified dataset* through a **profiling distance/algorithm**.
2. Compare the profiles of known users to users in the *anonymous dataset* to identify them using a **linking algorithm**.



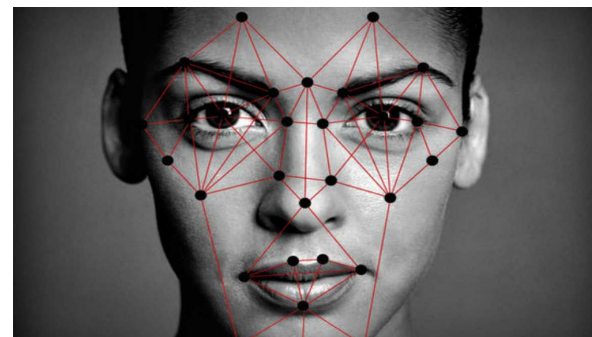
Profiling attacks: Face recognition

Face recognition: we want to extract features from your face that will not change too much with time, so as to **be able to recognise you over time**.

Many different techniques exist, e.g.:

- 1) *Ad hoc*, using anatomic features (size and position of the eyes, nose, ...)
- 2) *Eigenfaces*: PCA extraction of the distinguishing features of a face (using a large dataset of faces)
- 3) 3D matching techniques using metric geometry tools

For profiling attacks, we use the same idea: we want to extract time-independent features



Profiling attacks: location data

Attack model: assume you have access to last week's mobile phone metadata for a company, with names. The company publishes the data for this week, but without identifiers. Can you re-identify users

Profile: represent a person in the dataset by a *histograms of her locations*. This assumes that the place people go to and the time they spend there is similar from one week to another



User	Location		
	Dorm.	Rest.	Lib.
John	33%	33%	34%
Jill	70%	20%	10%
Mary	15%	60%	25%
Mike	15%	20%	65%

Profiling attacks: location data

Distance: Jensen-Shannon divergence

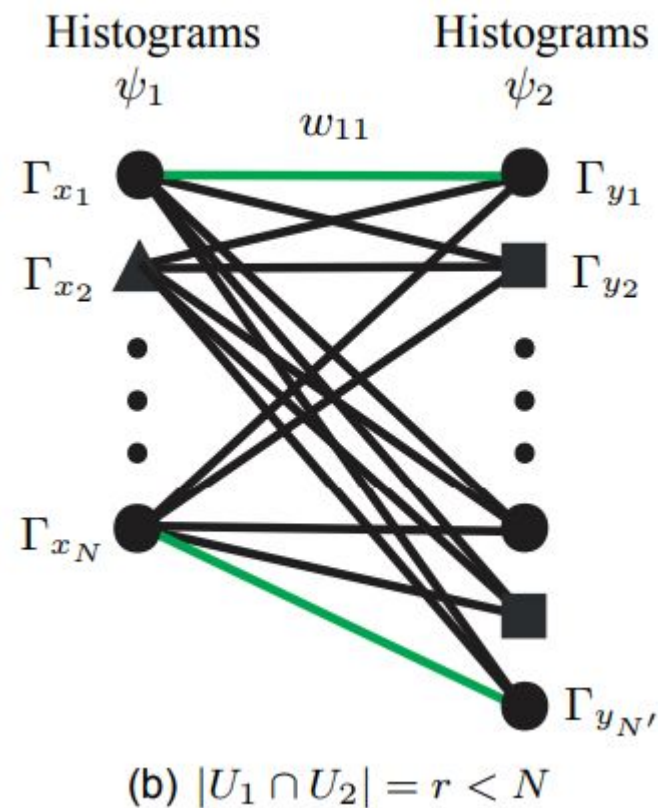
$$w_{ji} = D(\Gamma_{x_j} \parallel \frac{1}{2}(\Gamma_{x_j} + \Gamma_{y_i})) + D(\Gamma_{y_i} \parallel \frac{1}{2}(\Gamma_{x_j} + \Gamma_{y_i}))$$

Information theoretic metric that measures the (KL) distance to both histograms to their average.

Property: guarantees “optimal tradeoff” between type I and type II errors.

Matching algorithm

- 1) Compute histograms in both datasets
- 2) Compute the distance between each pair of histograms.
- 3) Use (again) the hungarian algorithm to find the minimal weight matching



Other distance metrics

Dot product: the dot product of the histograms (point by point), to measure how “aligned” the histograms are

$$w_{ji}^{\text{dot}} = \langle \Gamma_{x_j}, \Gamma_{y_i} \rangle = \sum_{l=1}^K \Gamma_{x_j}(l) \Gamma_{y_i}(l)$$

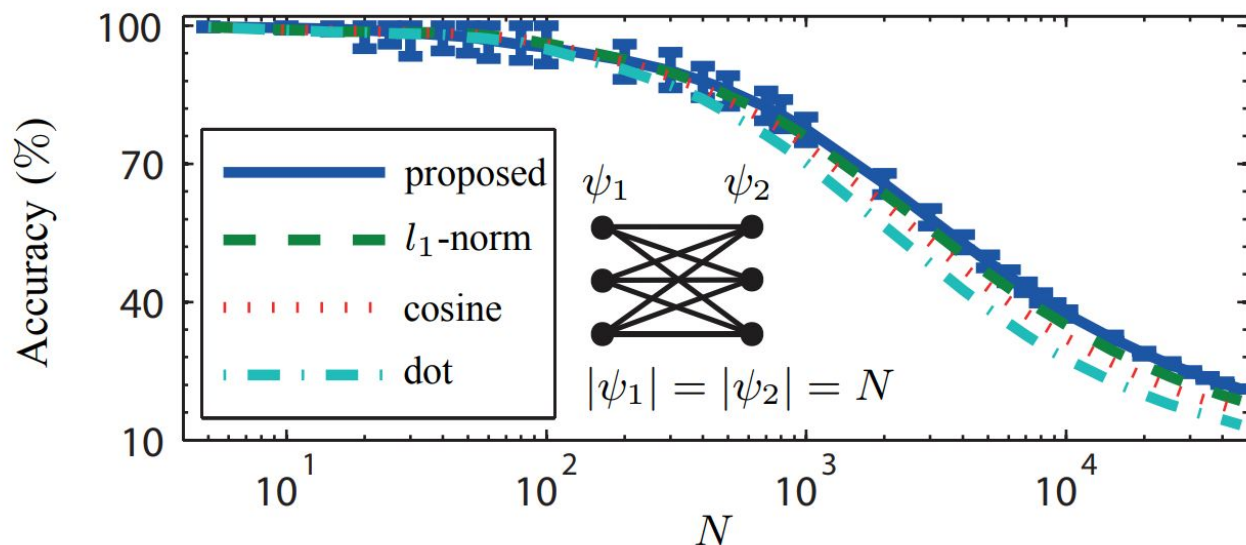
Cosine similarity: the dot product, but normalised by the norm of histograms, to allow for vectors of small norm (popular for sparse datasets):

$$w_{ji}^{\text{cos}} = 1 - \frac{\langle \Gamma_{x_j}, \Gamma_{y_i} \rangle}{\|\Gamma_{x_j}\|_2 \|\Gamma_{y_i}\|_2}$$

The L_1 distance (*Taxicab distance*): generic distance metric for vectors, popular in Machine Learning

$$w_{ji}^{l_1} = \|\Gamma_{x_j} - \Gamma_{y_i}\|_1 = \sum_{l=1}^K |\Gamma_{x_j}(l) - \Gamma_{y_i}(l)|$$

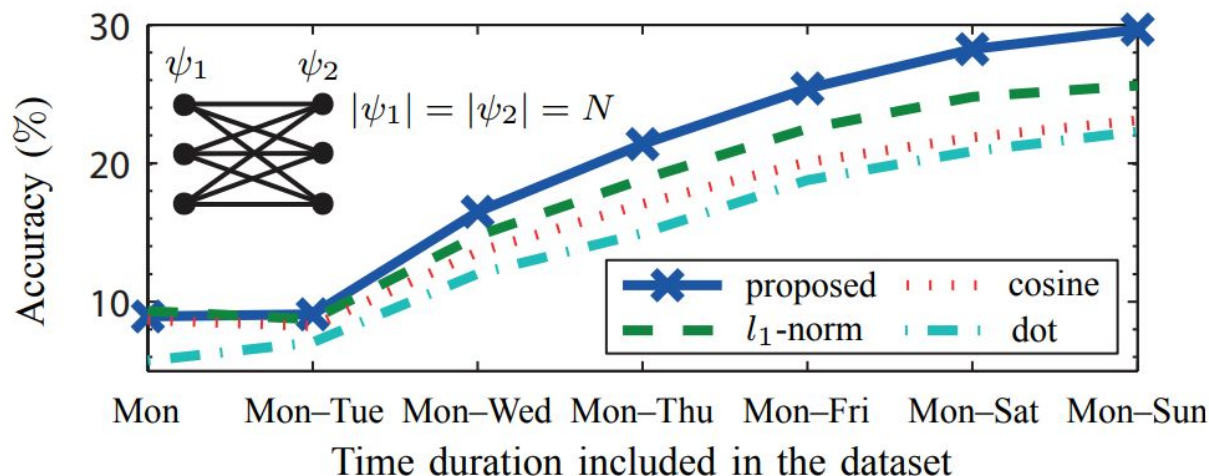
Profiling attacks: location data (results)



The accuracy (*proportion of matches found that are correct*, what was called *precision* in the other paper) is good for small datasets (up to 10^3 people), but **scale poorly** dropping to $\sim 40\%$ for 10^4 people.

Why would the accuracy of an attack drop so fast with size of population?

Profiling attacks: location data (results)



We can also study what happens when you have data for longer periods of time (using the same duration for both histograms).

The accuracy increases fast as we get more data (this is for 30K people). Considering longer time periods allows an attacker to better estimate user's behavior (here the places they spend time at) and capture more of their habits (what they do on the weekends etc).

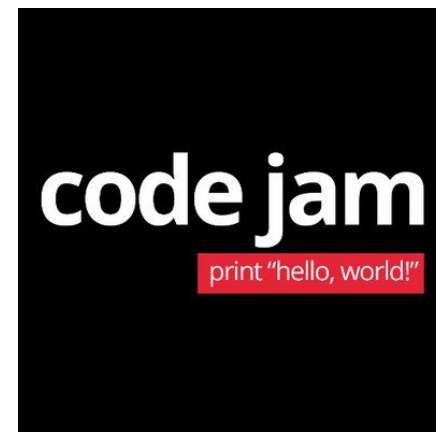
Profiling attacks: code authorship

Question: Is it possible to identify the author of a **binary** program, looking only at his style?

E.g. used to identify malware writers (e.g. The 2014 Sony attack, attributed to North Korea)

Dataset: Google Code Jam is a coding competition hosted by Google where *all programmers solve the same problems*.

They disassemble and decompile the machine code into C-like code which they then convert to abstract syntax trees (AST) of the code

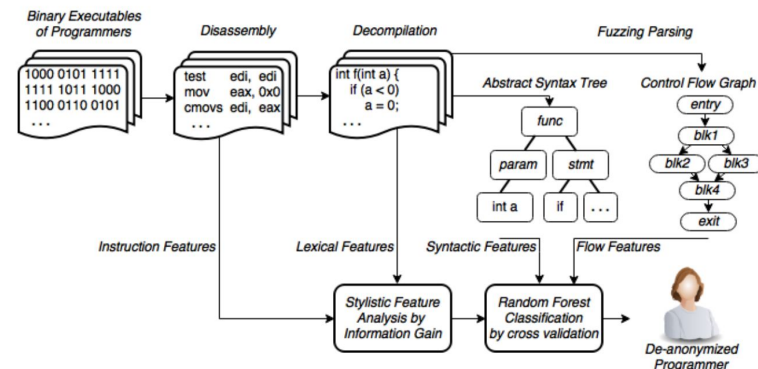


Why is it important to have all the programmers solve the same problem?

Profiling attacks: code authorship

First, they extract a lot (**700k**) of features from the binaries and, using **dimensionality reduction**, select 100 of them. It is computed in the same way for each programmer (*independently of the others*).

Second, they train a machine learning classifier (random forest) to distinguish between the different programmers using the 100 features. Each programmer is a class (and the classifier is built *for all programmers at the same time*)

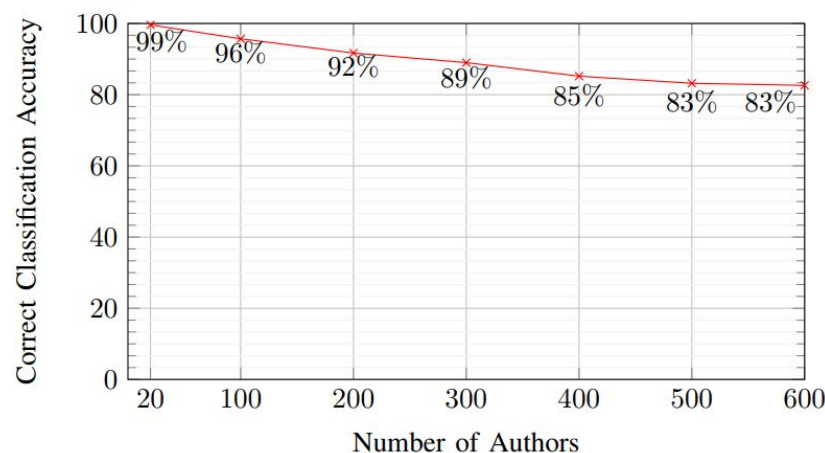


How scalable (in terms of number of programmers) is such an algorithm in your opinion?

Profiling attacks: code authorship (results)

For a small number of programmers (~600), the algorithm performs really well. This accuracy seems to decrease rather quickly with this number (sensing a theme here?)

→ The matching algorithm, however, is a classifier, that requires to be trained **on a specific set of programmers** and would not work on new programmers.



Big Data beyond transactions: Unstructured data

What is unstructured data?

Non-tabular, non-categorical data often text

For example:

- Tweets
- Emails
- Search queries
- Social graphs
- ...

But also browsing data (URLs), etc

This data can be very sensitive, yet useful
and difficult to anonymize



Unstructured data is highly identifiable

Aol.

Example: **The AOL leak.**

In 2006, AOL released the queries of 650k *pseudonymised* users to contribute to scientific research (and then retracted them immediately after being called out).

Reporters were able to identify *Thelma Arnold*, a 62-year-old widow from Georgia. Her queries referred to problems with old age, the area she lived in, and mentioned family members.

In web queries, it is **hard to predict which queries are going to be revealing of the user's identity**. This is a general theme in unstructured data.

A Face Is Exposed for AOL Searcher No. 4417749

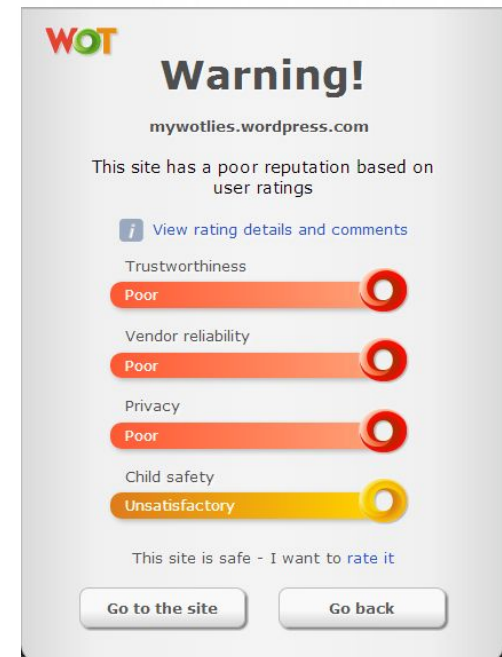
By MICHAEL BARBARO and TOM ZELLER Jr. AUG. 9, 2006

The Web of Trust controversy

Popular Browser extension, claims to provide “Safe Web Search & Browsing”, through a crowdsourced website reputation review, with more than 140M users.

Their privacy policy states that they store **all your browser data** in a “non-identifiable” way. They then **sell this data** to third-parties, according to an investigation by NDR, a German broadcasting station.

The data (some samples of which can be found online) was found to be **trivially identifiable**, and containing email addresses, user illnesses, sexual orientation, drug usage, ...



Questions?