Preventing Keystroke Based Identification in Open Data Sets

Juho Leinonen, Petri Ihantola, Arto Hellas


DOI = https://doi.org/10.1145/3051457.3051458
In Brief

- Detailed data
- Open data
- Privacy?
What kind of data do we collect on our courses?
Heinonen et al. "Using CodeBrowser to seek differences between novice programmers." *SIGCSE '14*
Raise your hand if you would like to have this programming process data openly available!
What about privacy?
Privacy

- What could happen if we shared this data openly?

- Netflix Prize “anonymized” data set linked with the Internet Movie Database records [1]
  - Political preferences, hints about sexual orientation

Privacy

- Combining our data with other data sets

- E.g. potential advertisers know this person took our MOOC → advertise other MOOCs to them

- Inferring emotional states [1]?

Privacy solutions

- Delete information?
- Reduces quality of the data
- Modify data?
Preventing Keystroke Based Identification (whilst retaining value in the data)
Preventing Keystroke Based **Identification**
(whilst retaining value in the data)
Typing profiles

- Keystrokes & Timestamps → Typing profile
  - Contains average character pair latencies
  - Top 10 = correctly identified

h → e → l → l → o
22 → 75 → 107 → 120 → 138

Average latencies:
he = 53
el = 32
ll = 13
lo = 18

Longi et al. "Identification of programmers from typing patterns." Koli Calling ‘15
Preventing Keystroke Based Identification (whilst retaining value in the data)
Inferring programming experience from typing

- Experience or no experience?

- Best features include: i+ || {} tr ru ue

Leinonen et al. "Automatic inference of programming performance and experience from typing patterns." SIGCSE ’16
Research Question:

Can we modify the typing profiles so that:

1) Identification is unreliable

2) Programming experience can still be inferred
Methodology
Methodology

- In search for the sweet spot: identification unreliable, programming experience inference possible

- Two methods of modifying typing profiles
  - Rounding \([1]\) average latencies
  - Bucketing average latencies

\([1]\) Similar to generalization
Rounding method
Rounding method

- Round the values to nearest $x$ milliseconds
  - E.g. all values in data are rounded to nearest 100 ms
  - Leads into “buckets”

- 0–50 ms $\rightarrow$ 0 ms
- 50–150 ms $\rightarrow$ 100 ms
- 150–250 ms $\rightarrow$ 200 ms
- ...
Bucketing method
Bucketing method

- Distribute values into buckets that have equal ranges.
- Similar to the rounding method, except the first bucket is the same size as the rest.
- 100 ms buckets:
  - 0–100 ms → 100 ms
  - 100–200 ms → 200 ms
  - 200–300 ms → 300 ms
  - …
Results
Bucketing
Identification vs programming experience inference

The diagram shows the identification and classification accuracy percentage as a function of bucket size in milliseconds. Different methods are compared, including Identification, Bayes Net, Random Forest, and Majority Classifier. The accuracy decreases as the bucket size increases.
Sweet spot?
Rounding
Identification vs programming experience inference
Identification vs programming experience inference
Sweet spot?
Discussion
Rounding identification accuracy

![Graph showing rounding identification accuracy over time with two lines representing 2014 and 2015.](image)
Rounding identification accuracy

[Graph showing identification accuracy percentage over rounded values in milliseconds for two years, 2014 and 2015, with a notable peak in 2014 near 300 milliseconds.]
Course with different students and content
Hypothesis

(not in the paper)
Weird effect seen with the rounding method

Distribution of average latencies
Rounding to 170 ms (low identification)

Rounding to 340 ms (high identification)
Limitations
Limitations

- Analyzed single context -- external validity?
- Other identifiable information likely remains (variable names, etc.) -- internal validity?
Conclusions
Conclusions

- Identified approaches for anonymizing fine-grained programming process data
  - Starting point for releasing open keystroke data
  - Sweet spot for anonymization related to methodology
  - Bigger anonymization not necessarily better
  - Need to study other identifiers
Conclusions

- Identified approaches for anonymizing fine-grained programming process data
  - Starting point for releasing open keystroke data
  - Sweet spot for anonymization related to methodology
  - Bigger anonymization not necessarily better
  - Need to study other identifiers

Thank you!
Conclusions

- Identified approaches for anonymizing fine-grained programming process data
- Starting point for releasing open keystroke data
- Sweet spot for anonymization related to methodology
- Bigger anonymization not necessarily better
- Need to study other identifiers

Thank you!
Questions?
References

Thank you!