Università di Genova
DIBRIS DIPARTIMENTO DI INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

(k, P)-Anonymity

Data Protection & Privacy (90538) - a.y. 2020/21

Group: DPP9

Federico Minutoli
Sofia Bagnato
Matteo Ghirardelli
Gianvito Losapio

Table of Contents

Table of Contents

Time-series data

- Sequences of observations captured at regular intervals, indexed by the time instant of each observation

- Usually mined for forecasting

- Support for two types of complex queries:
  - Range queries on attribute values
  - Pattern similarity queries

Time-series data

A generic record in a database of time series, *T*, should contain:

- ➔ a unique identifier, *Id*;
- ➔ a set of quasi-identifier (*QI*) attributes at *n* different but mainly consecutive time instants, denoted by *QI = {A1, A2, …, An}*
- ➔ a set of sensitive attributes, *SD*.

Id	QI						SD
Name	2005	2006	2007	2008	2009	2010	2011(A_S)
Alice	170	175	188	197	213	221	200
Bob	145	157	165	177	204	196	180
Cathy	176	181	147	134	125	112	160
David	98	120	125	132	151	161	110
Jane	117	107	87	74	51	56	85
Lily	32	54	59	67	96	101	90
Mary	88	93	56	43	20	25	55
Steve	71	63	47	38	43	20	46

Time-series data: Challenges

- ➔ “Privacy protection in the publication of time series is a challenging topic mostly due to the complex nature of the data and the way that they are used.”

- ➔ High dimensionality makes boundary between QI and SD attributes much more difficult to identify.

- ➔ Adversary background knowledge is impossible to model.

Time-series data: Challenges

- ➔ Both types of complex queries should be accountable even after the anonymization procedure. In order to do that:
  - ◆ Patterns over time should be preserved
  - ◆ Statistical properties of time series should be preserved

- ➔ Common *generalization-based* anonymization methodologies, such as *k-anonymity*, may perform sub-optimally

Related work

Existing partial information hiding approaches can be divided into two disjoint categories:

- *Perturbation-based approaches*, which protect data by adding noises according to some kind of distribution to make the perturbed data have several common characteristics with the original data

- *Partition-based approaches*, which first divide tuples of database into disjoint groups and then release some general information out of each group, i.e., *k-anonymity*, condensation, etc.

Related work

“No previous work has adequately addressed the anonymization of time series to answer the most frequently used (range and similarity) queries in the published database.” [1]

[1] L. Shou et al., “Supporting Pattern-preserving Anonymization for Time-series Data” in IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 4, pp. 877-892, April 2013, doi: 10.1109/TKDE.2011.249

Table of Contents

Why not k-anonymity

k-anonymity

- ➔ Common solution to prevent linkage attacks
- ➔ Statistical properties of time series data are kept, at the expenses of patterns similarity

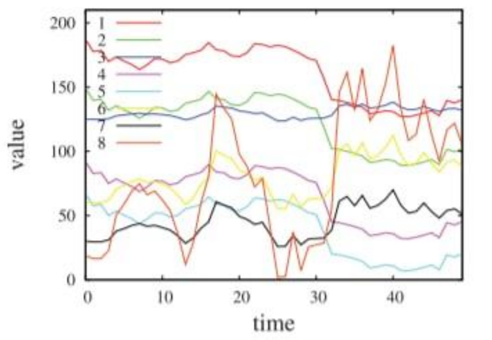
Why not k-anonymity

k-anonymity

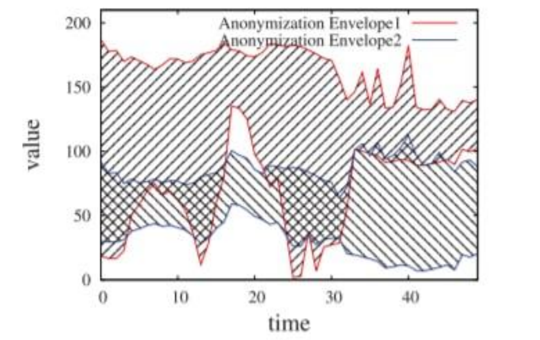
- ➔ Common solution to prevent linkage attacks
- ➔ Statistical properties of time series data are kept, at the expenses of patterns similarity

Not good enough!

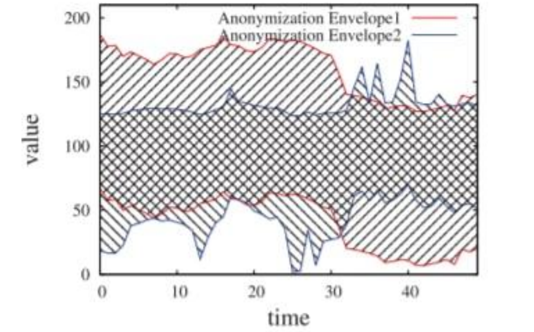
Why not k-anonymity



(a) Micro data



(b) Generalization result of conventional 4-anonymity. Group 1 contains 1,2,3,8, while group 2 contains 4,5,6,7.



(c) Generalization result of conventional 4-anonymity based on pattern similarity. Group 1 contains 1,2,4,5, while group 2 contains 3,6,7,8.

Generalization of conventional 4-anonymity based on Euclidean distance (b) vs pattern similarity (c)

(k, P)-anonymity: Why

k-anonymity	(k, P)-anonymity
➔ Common solution to prevent linkage attacks	➔ Can prevent both linkage and pattern disclosure attacks
➔ Statistical properties of time series data are kept, at the expenses of patterns similarity	➔ Generalization of k-anonymity: k-anonymity + P-anonymity
↓ Not good enough!	➔ Patterns similarity of time series are well preserved

(k, P)-anonymity: What

"Anonymized data publishing should jointly present in different format a set of groups with minimum size *k* having the same anonymization envelope (AE), which are further divided into groups with minimum size *P* having the same pattern representation (PR), and those PRs"

Dual anonymization: k and P

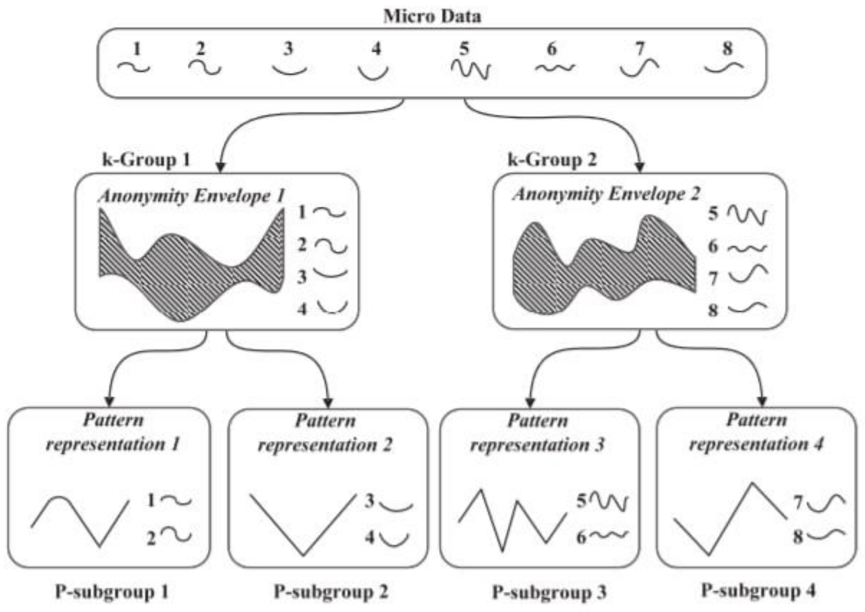


Fig. 2. The k-groups and P-subgroups of (k,P)-anonymity.

Dual anonymization: k and P

- k-requirement: each AE must appear at least *k* times

- P-requirement: for each k-group *G*, and for each record *r* in *G*, there must be at least *P - 1* other records in *G* having the same *PR*, i.e., *PR[r]*

k, P and ...?

- ➔ k-anonymity ensures traditional protection against re-identification based on single QI attribute knowledge.

- ➔ P-anonymity ensures protection against attacks based on pattern knowledge.

- ➔ Additional privacy can be granted by enforcing stricter flavours of k-anonymity, such as *l-diversity*, in order to prevent homogeneity attacks and make background knowledge less of running force.

Pattern Representation (PR)

- Feature

$$f : (A_1, \dots, A_n) \rightarrow Y$$

- Pattern (dimensionality reduction + domain of range/similarity queries)

$$p(r) = [f_1, f_2, \dots, f_m]$$

- Pattern Representation

$$\mathcal{M}(r) = \Phi\big(p(r)\big)$$

$$\hat{\Phi}(\mathcal{M}(r)) = \hat{p}(r) \simeq p(r)$$

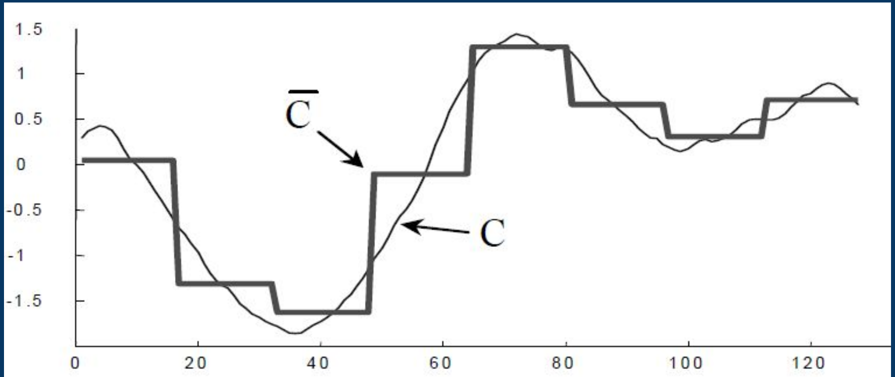
SAX

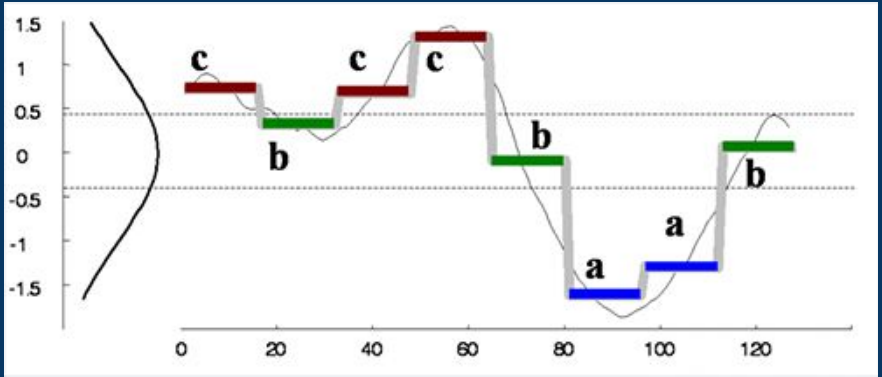
→ SAX is a popular symbolic representation for time series (2002)

$$\mathcal{M}(r) = \Phi\big(p(r)\big)$$

$$p(r) = PAA(r)$$

$$\Phi(p) = \mathrm{num2sym}(p)$$





$$\bar{C} = [\bar{c_1}, \bar{c_2}, \ldots, \bar{c_w}]$$

$$\underbrace{[\beta_0;\beta_1)}_{\alpha_1}, \quad \underbrace{[\beta_1;\beta_2)}_{\alpha_2}, \quad \ldots, \quad \underbrace{[\beta_{a-1};\beta_l)}_{\alpha_l}$$

$$\bar{c_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} z_j$$

$$\int_{\beta_i}^{\beta_{i+1}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}\, dx = \frac{1}{l}$$

Table of Contents

Normalized certainty penalty (NCP)

- Useful metric to measure the uncertainty caused by data generalization.

- Given an attribute Ai, the NCP is computed as $NCP_{A_i}(t) = \frac{|z_i - y_i|}{|A_i|}$, with $A_i = max\{T.A_i\} - min\{T.A_i\}$ the range of all tuples on attribute Ai.

- Once defined for one attribute and record, it can be generalized to whole table as $NCP(T) = \sum_t w_t \times NCP(t)$ with w being an optional weight.

Instant value loss (VL)

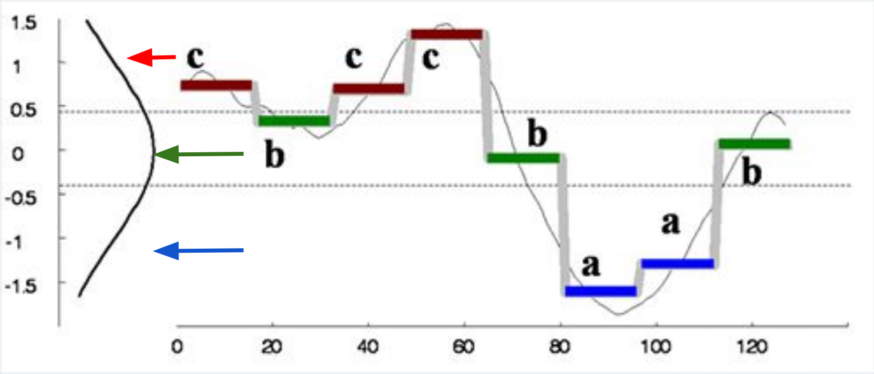
- Loss measure based on the anonymization envelope of each k-group.

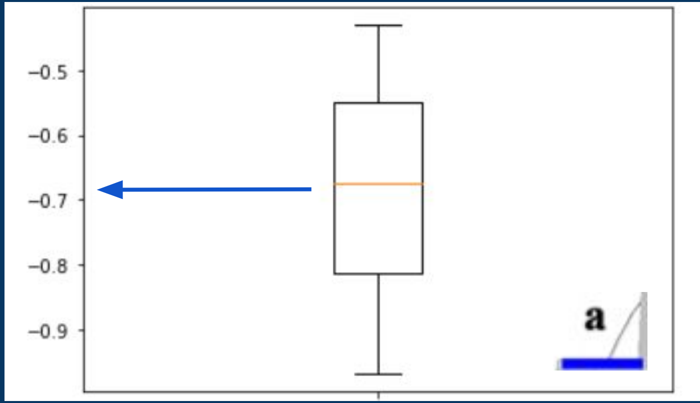
- Given anonymization envelope lower bounds $\left(r_1^-, r_2^-, \ldots r_n^-\right)$ and upper bounds $\left(r_1^+, r_2^+, \ldots r_n^+\right)$ we can define the VL of a record Q belonging to it as $VL(Q) = \sqrt{\sum_{i=1}^{n} (r_i^+ - r_i^-)^2 / n}$.

- The total for the whole table is obtained by summing up the VL for each record.

Pattern loss (PL)

$$\hat{\Phi}(\mathcal{M}(r)) = \hat{p}(r) \simeq p(r)$$





$$\ell(p, \hat{p}) = 1 - \frac{p \cdot \hat{p}}{||p||\ ||\hat{p}||}$$

➔ SAX represents a natural choice for the Pattern Representation (PR):

- easy to understand
- the accuracy can be tuned (for the P-requirement and the needs of minimizing pattern loss)
- general purpose (for all different usages of the published data)

Table of Contents

l - diversity

- L-diversity is an additional privacy measure which can be enforced after the (K, P) anonymization.
- Given a P-group and a sensitive attribute from a time series belonging to it, l-diversity is respected if and only if the sensitive attribute appears a maximum of $\frac{|P\_group|}{l}$ times.
- This makes sure that no information about original users can be inferred even by the anonymized dataset, since if every P-group record has the same sensitive data value then information about those belonging to that group can be easily gathered.

l - diversity

- ➔ L-diversity is enforced by following a perturbation-based procedure.

- ➔ Each sensitive data not satisfying l-diversity is perturbed.

- ➔ Each time we perturbate a value, a check is done to see if the change might cause issue with actual pre-existing sensitive values, in which case if too many issues arise an iterative method to widen the perturbation search space is employed.

l - diversity

- ➔ Due to not having a utility metric for sensitive data, only qualitative reasoning was done and no plots were generated.

- ➔ Various l values were tried (2, 3 and 4) with small p values (around 6).
- ➔ The bigger the l, the bigger the number of records that needed alteration.

- ➔ In any case, small values of P made the amount of needed perturbation smaller, as the applied perturbation in any case is smaller than P.

Table of Contents

Implementation challenges

- Existing code reuse
  - Reusing the code seen during class proved to be both an advantage as well as an added difficulty;
  - Program logic had to be re-adapted to our ideas and integrated into the new program. Adapting it proved to be quite the challenge.

- Changed functions for node splitting, finding max NCP, top-down clustering postprocessing (to turn bad groups into good ones to avoid infinite recursion by selecting left or right neighbor)

Naive algorithm

➔ **Top-down approach:**

- **First**, it produces k-subgroups from the whole table with a **top-down clustering** procedure;
- **Then**, it splits these groups into P-subgroups via the **create-tree** procedure;
- The **create_tree** procedure generates a list of good leaves, each representing a P-subgroup.

➔ **Computational complexity** of O(max-level * |T| + |T|^2).

Naive algorithm - top-down clustering

Input: a table T, parameter k, weights of attributes, hierarchies on categorical attributes;
Output: a k-anonymous table T';
Method:

```
1: IF |T| ≤ k THEN RETURN;
2: ELSE {
3:     partition T into two exclusive subsets T1 and T2 such
       that T1 and T2 are more local than T, and either T1
       or T2 have at least k tuples;
4:     IF |T1| > k THEN recursively partition T1;
5:     IF |T2| > k THEN recursively partition T2;
   }
6: adjust the groups so that each group has at least k tuples;
```

Naive algorithm - create-tree, node splitting phase

Algorithm 1: **Node splitting**

```
Data: tree node N, P, max-level
begin
    if N.size<P then
        N.label = bad-leaf;
    if N.level==max-level then
        N.label = good-leaf;
    if P ≤N.size<2 ∗ P then
        N.label = good-leaf;
        Maximize N.level without node split;
    else
        if N can be split then
            if total size of all TB-nodes ≥ P then
                generate child_merge;
                child_merge.level= N.level;
                level of all TG-nodes is N.level + 1;
            else
                level of all child nodes is N.level + 1;
        else
            N.label = good-leaf;
end
```

Naive algorithm - create-tree, postprocessing phase

➔ Bad leaves are sorted in ascending order by size.

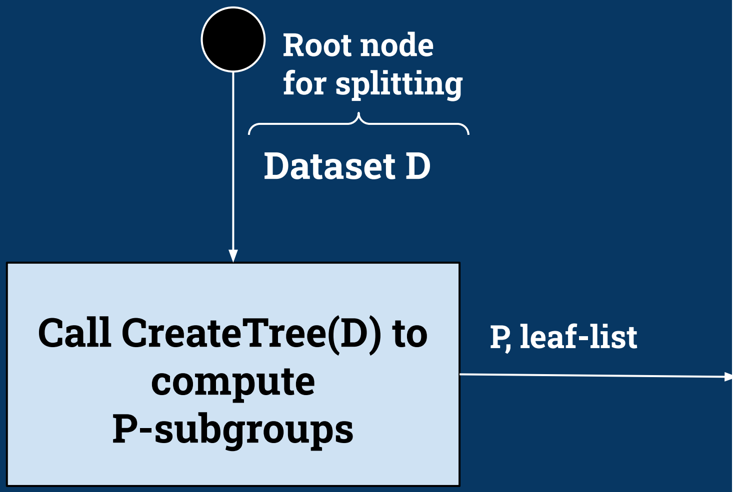
➔ Each bad leaf is taken and merged into a good leaf having the most similar pattern representation; pattern representation of the leaf will not change.

➔ Ties are broken by choosing the smallest good leaf.

KAPRA algorithm

- **Improvement over** the **naive** algorithm
  - **Finer** pattern **representations**
  - **Overcomes the search space limitations** of node splitting
- **Bottom-up** approach:
  - **First**, it produces **p-subgroups by splitting** the **entire dataset**
  - **Then**, it **forms k-groups from p-subgroups**

KAPRA algorithm: Step 1-2



No post-processing performed! Computational complexity is O(maxlevel * |T|).

Computational complexity is O(max-bad-level * num-bad-leaves).

Algorithm 2: Recycle bad-leaves

```
Data: P, leaf-list, current-level, max-bad-level
Result: P-subgroup list
begin
    current-level = max-bad-level;
    while sum of all bad leaves' size ≥ P do
        if any bad leaves can merge then
            Merge them to a new node leaf-merge;
            if leaf-merge.size≥ P then
                leaf-merge.label= good-leaf;
            else
                leaf-merge.label= bad-leaf;
        current-level − −;
    Suppress all time-series contained in bad leaves;
end
```

KAPRA algorithm: Step 3

Computational complexity is O(|PGL|^2).

Algorithm 3: **Group formation**

Data: PGL, k, P

Result: Group list GL

begin
- **for** *each P-subgroup that size* $\geq 2 * P$ **do**
  - Split it by top-down clustering;
- **if** *any P-subgroup that size* $\geq k$ **then**
  - Add it into GL and remove it from PGL;
- **while** $|PGL| \geq k$ **do**
  - Find s_1 and $G = s_1$;
  - **while** $|G| < k$ **do**
    - Find s_{min} and add s_{min} into G;
  - Remove all P-subgroups in G from PGL and put G in GL;
- **for** *each remaining P-subgroup* s' **do**
  - Find corresponding G' and add s' into G';

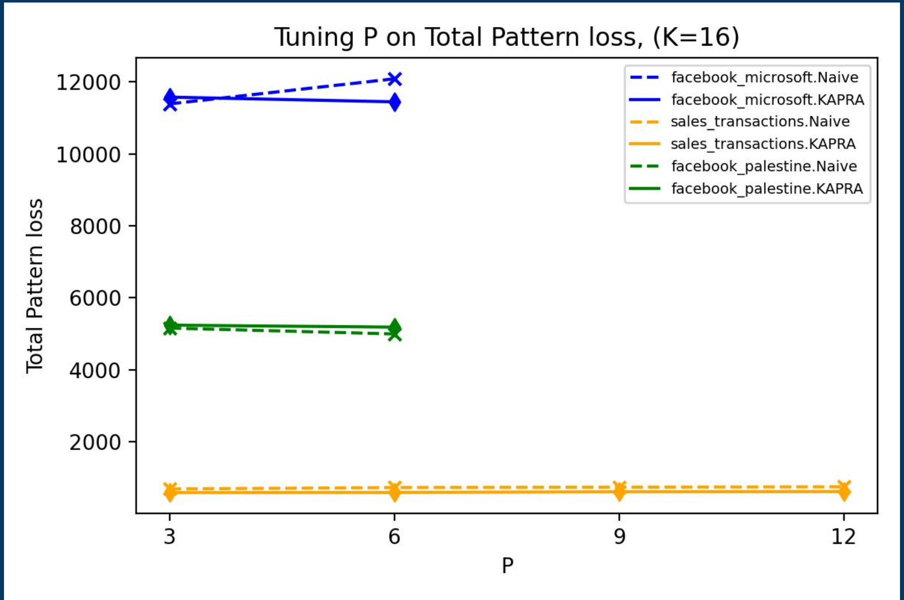
end

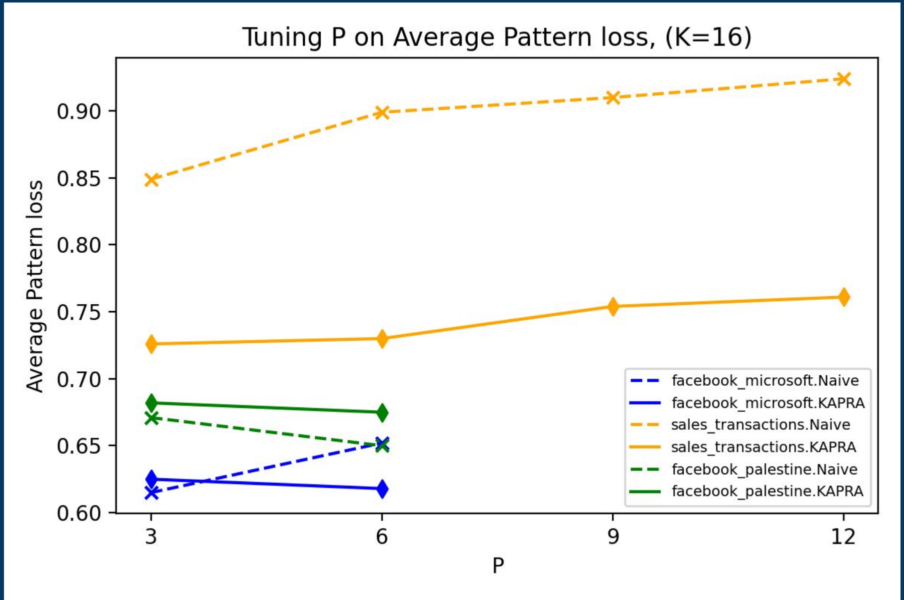
Table of Contents

Datasets

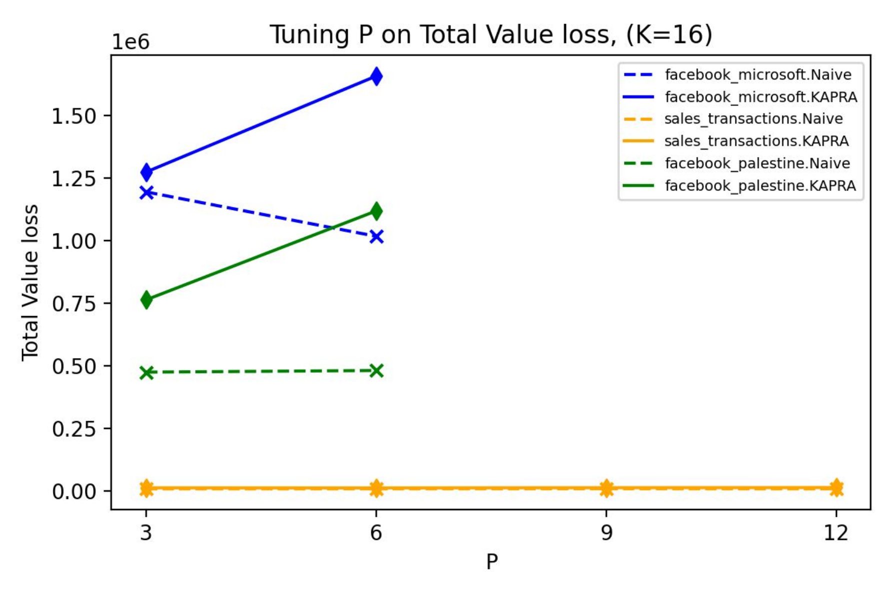
- **Weekly purchased quantities of 800 products over 52 weeks,** contained in **Sales_transactions_dataset_weekly.csv**
  - Can be retrieved at https://archive.ics.uci.edu/ml/datasets/sales_transactions_dataset_weekly
- **Large data sets of news items and their respective social feedback on** the **Facebook** platform:
  - **Facebook_microsoft.csv: Microsoft products-related data**
  - **Facebook_palestine.csv: Palestinian terrorism-related data**
  - Can be retrieved at https://archive.ics.uci.edu/ml/datasets/News+Popularity+in+Multiple+Social+Media+Platforms
- **Each product or news item** (a row) **is** associated with **a time series**

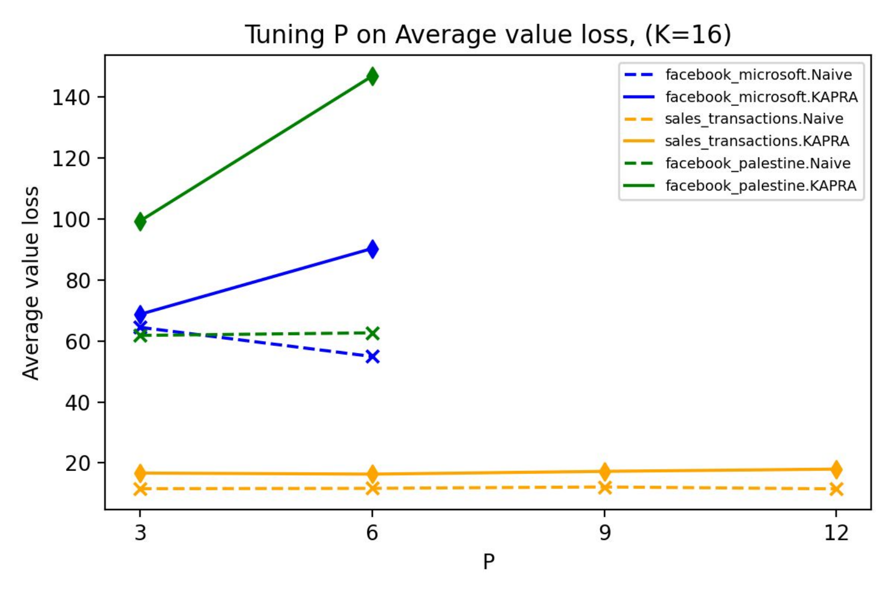
Pattern loss experiment results: fixed K, increasing P



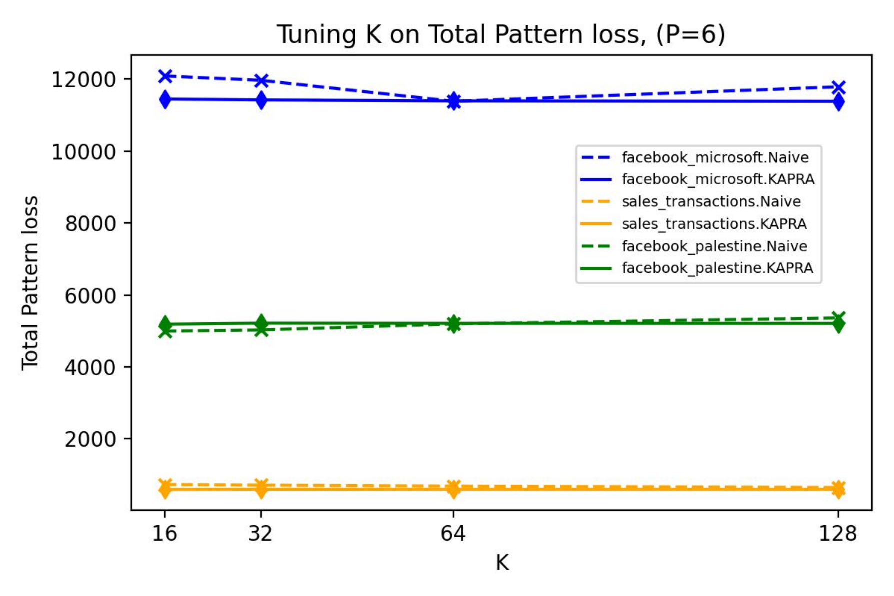


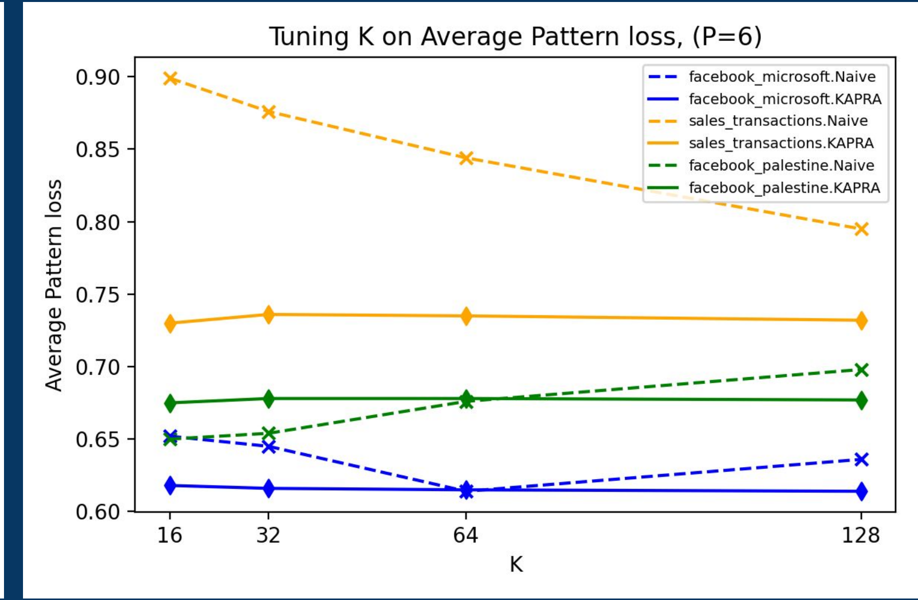
Value loss experiment results: fixed K, increasing P



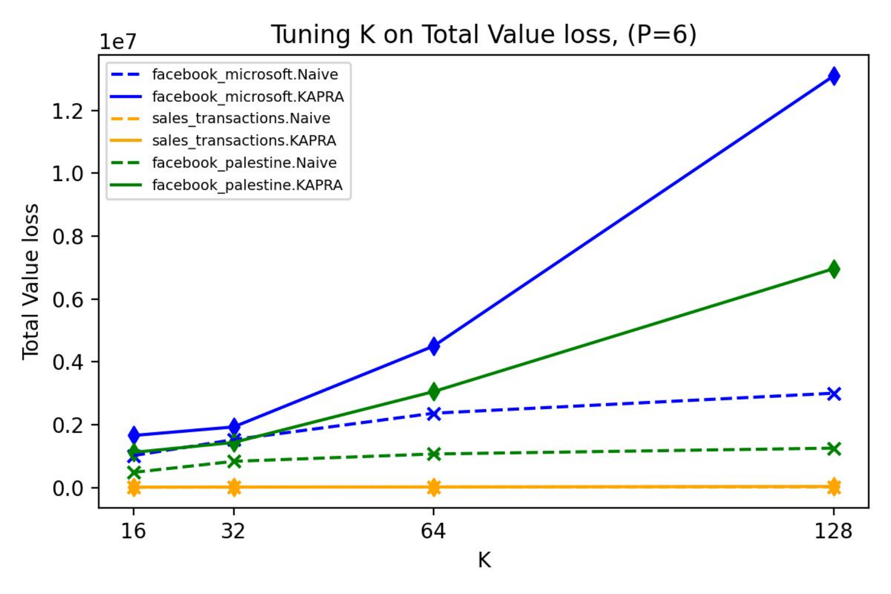


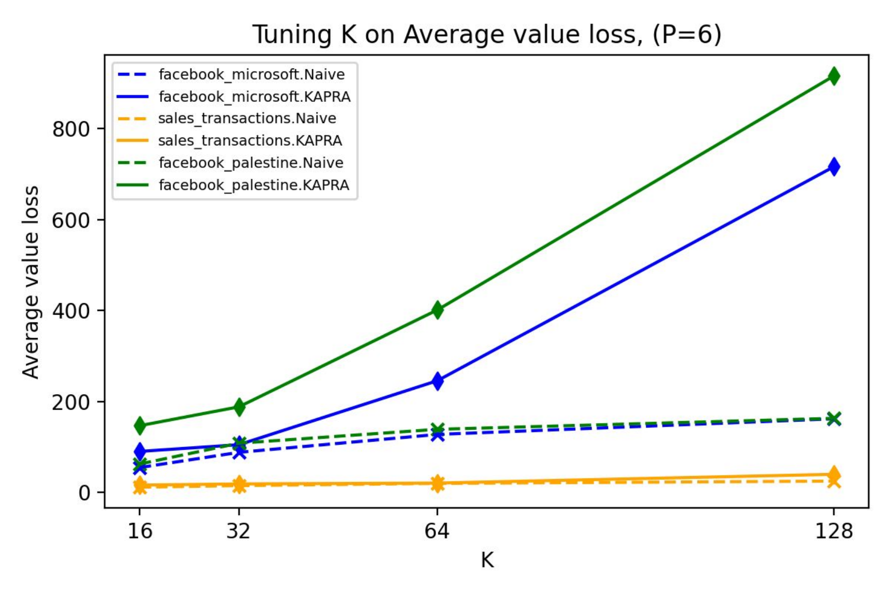
Pattern loss experiment results: fixed P, increasing K



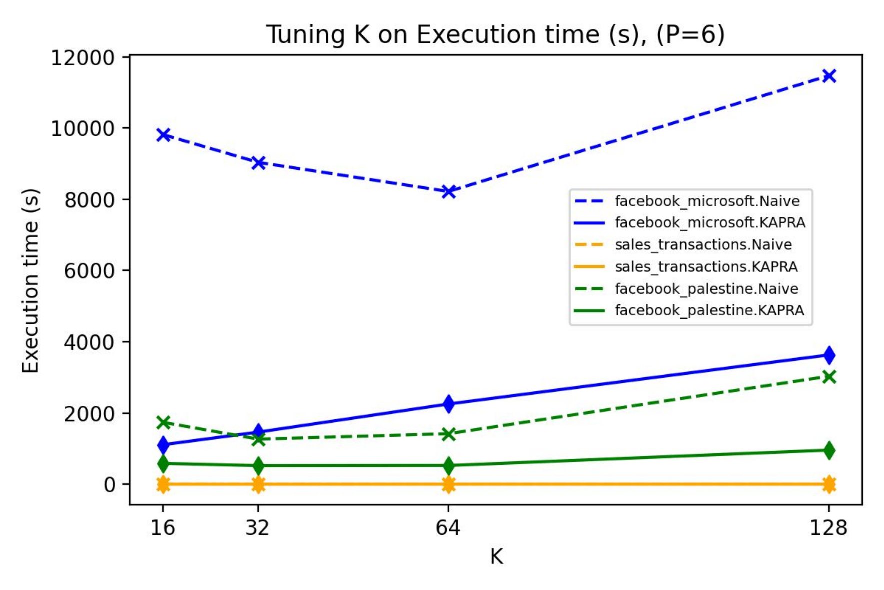


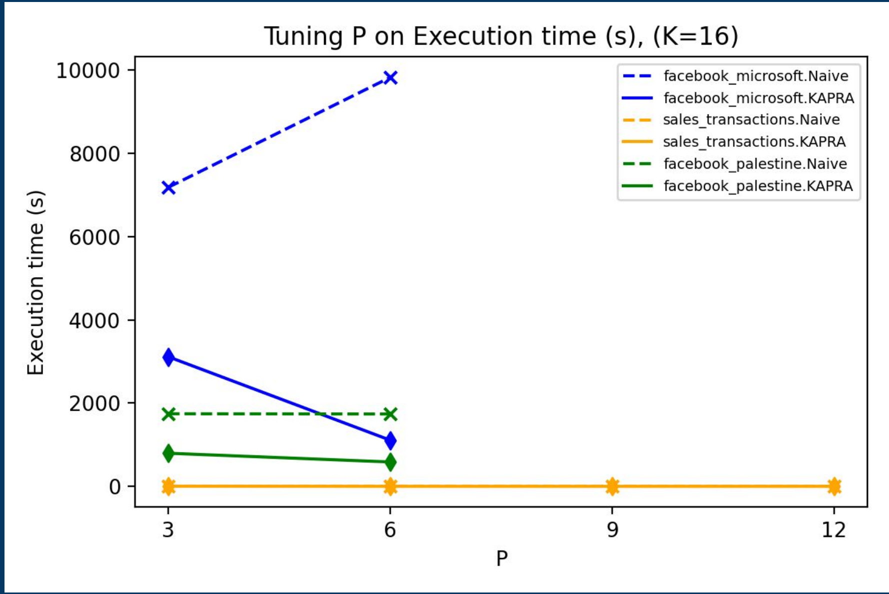
Value loss experiment results: fixed P, increasing K





Anonymization runtime





KAPRA!   KAPRA!   KAPRA!

Thanks for the attention!