EPFL

Private k-NN Graph Construction in Decentralized Recommender Systems

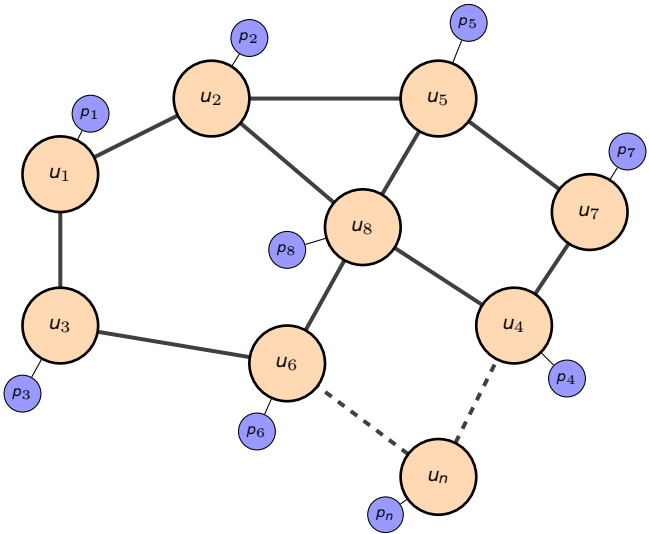
Yassine Boukhari

Decentralized Recommender Systems

- Recommender Systems struggle to handle **large** amounts of data/users.
- Servers can't be trusted with the **private** data of users.

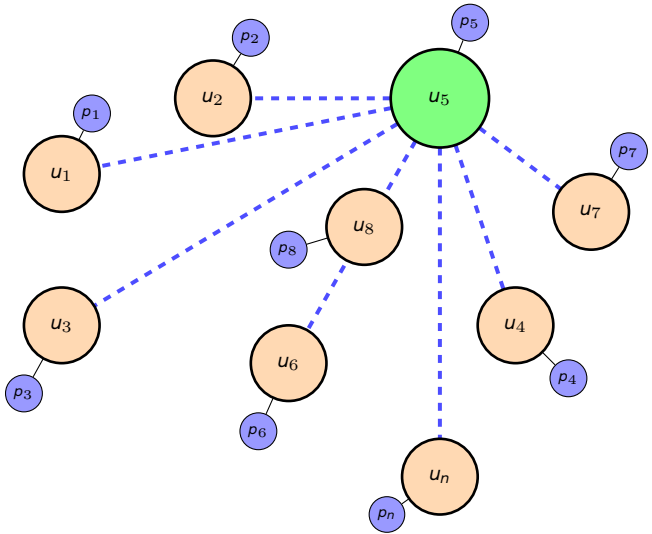
The need for scalability & privacy triggered a transition towards more decentralized recommenders.

User-based Collaborative Filtering

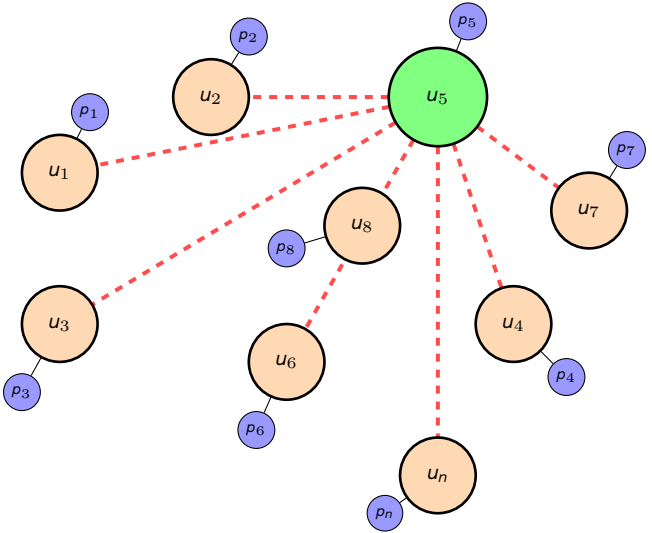


k-NN Graph Construction

$$\textbf{Similarity} = \frac{|\mathbf{p_i} \cap \mathbf{p_j}|}{|\mathbf{p_i} \cup \mathbf{p_j}|}$$



Computing the similarity requires revealing the user profile !



Build k-NN Graph + **Preserve User Privacy**

Private Set Intersection Cardinality

- Server, Client, every party holds a set.
- **Goal:** client learns the intersection cardinality and nothing else.
- **Semi Honest Model:** privacy and correctness.
- **Malicious Adversaries:** privacy.
- **Linear** communication/complexity, 2 messages.

Decentralized Private Set Intersection Cardinality

- ~~Server~~, ~~Client~~, **P2P**, every party holds a set.
- **Goal:** **both parties** learn the intersection cardinality and nothing else.
- **Semi Honest Model:** privacy and correctness.
- **Malicious Adversaries:** privacy.
- **Linear** communication/complexity, **3** messages.

Decentralized Private Set Intersection Cardinality

- ~~Server~~, ~~Client~~, **P2P**, every party holds a set.
- **Goal:** **both parties** learn the intersection cardinality and nothing else.
- **Semi Honest Model:** privacy and correctness.
- **Malicious Adversaries:** privacy.
- **Linear** communication/complexity, **3** messages.

→ **Private Jaccard similarity** → **Private graph construction**

In this Project

- Implement privacy preserving k-NN graph construction with PSI-CA.
- Benchmark the system.
- Study the overheads of achieving privacy.

To achieve these goals

- **fpsica:** High performance C++/Python library for PSI-CA.
  - `NIST P-256` curve, `SHA-256` hashes.
- **decentralizepy :**
  - Developed a skeleton to run interactive protocols with minimal changes.
  - Extended with KNN and PSICA-KNN nodes.
- **benchmarks:** fpsica, convergence, communication, computation.

Benchmarking fpsica

- Benchmark against **petlib**.
- Measure execution time for every phase of the protocol.
- Scale input size.

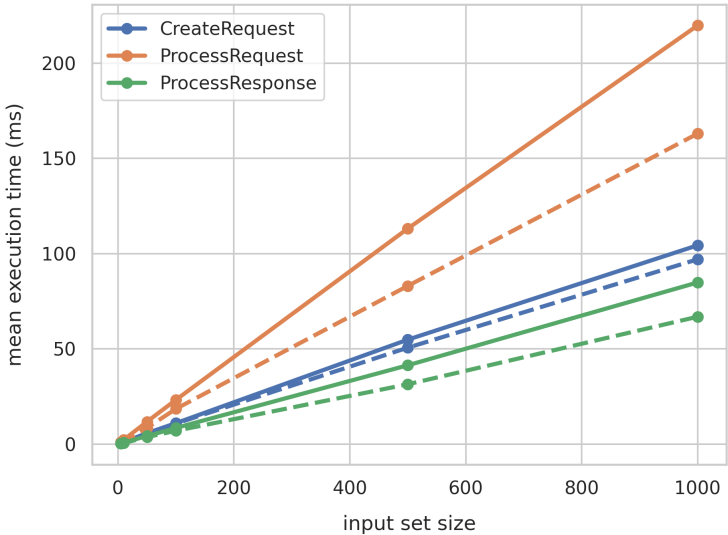


Figure: fpsica vs petlib

Convergence of the k-NN algorithm

- Measure the k-NN graph quality for every round.
- Different values of k, scale the graph size until 128 nodes.
- ≥ 0.8 after $log_2(\#nodes)$ rounds.

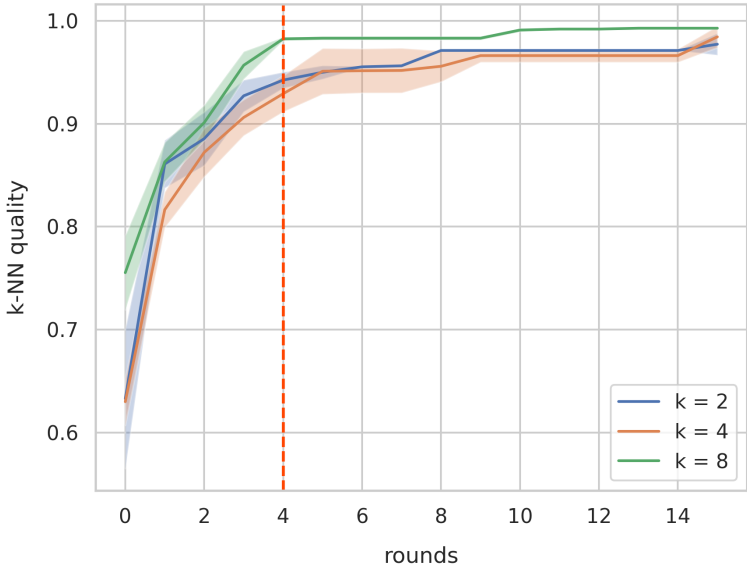


Figure: k-NN graph quality over rounds for 16 nodes

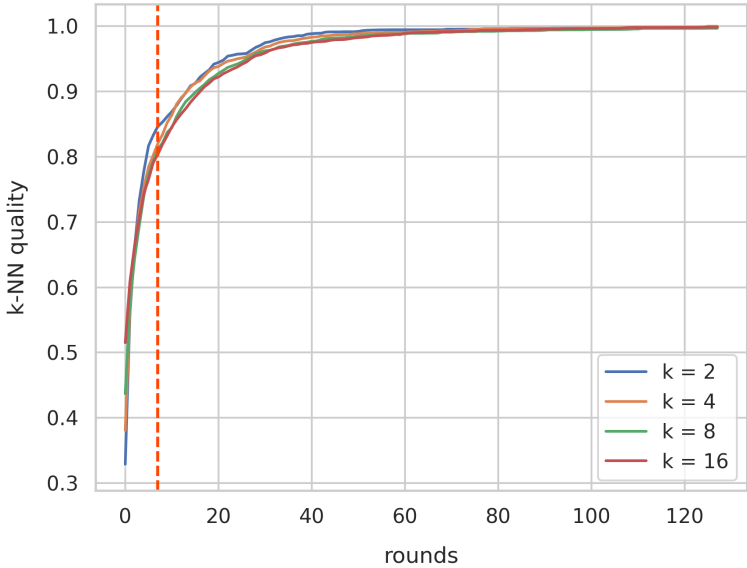


Figure: k-NN graph quality over rounds for 128 nodes

Communication overhead

- Run the graph construction for (# nodes) rounds.
- Measure the total number of exchanged bytes.
- Overhead compared to regular k-NN.

# nodes \ k	2	4	8	16
8	3.58	3.62	–	–
16	3.45	3.76	3.69	–
32	3.34	3.36	3.42	–
64	3.40	3.32	3.31	3.45
128	3.27	3.31	3.27	3.27

Table: Communication Overhead of PSI-CA k-NN

Computation overhead

- Run the graph construction for (# nodes) rounds.
- Measure the average time per k-NN round.

# nodes \ k	2	4	8	16
8	3.92	3.66	-	-
16	3.23	3.53	3.54	-
32	3.46	3.47	3.52	-
64	3.30	3.48	3.27	3.44

Table: Added latency of PSI-CA k-NN

# nodes \ k	2	4	8	16
8	3.92	3.66	-	-
16	3.23	3.53	3.54	-
32	3.46	3.47	3.52	-
64	3.30	3.48	3.27	3.44

Table: Added latency of PSI-CA k-NN

→ **Similar results in poor network conditions**

- Privacy is expensive.
- Overhead: protocol requires extra messages.
- Aim for protocols with low communication costs or non interactive protocols.

Questions