Real-time Collaborative Filtering Recommender Systems

Huizhi Liang, Haoran Du, Qing Wang

Presenter: Qing Wang

Research School of Computer Science
The Australian National University
Australia

Partially funded by the Australian Research Council (ARC), Veda Advantage, and Funnelback Pty. Ltd., under Linkage Project.

Introduction – Recommender Systems

- **Applications**
  - Predict topics that would trend on Twitter
  - Predict fluctuations in the prices of Bitcoin
  - ...

Introduction – Recommender Systems

- **Applications**
  - Predict topics that would trend on Twitter
  - Predict fluctuations in the prices of Bitcoin
  - . . .

- **Common techniques**
  - Collaborative filtering

    i.e., use the ratings of users and items
  - Content-based filtering:

    i.e., use the features of users and items
  - Hybrid techniques

    i.e., combine the above two to overcome their limitations

Collaborative Filtering

- Coined by Goldberg et al. in Tapestry (1992): "people collaborate to help one another perform filtering by ..."

Collaborative Filtering

- Coined by Goldberg et al. in Tapestry (1992): "people collaborate to help one another perform filtering by ..."
- **Assumption**
  - If two users act on n items similarly (e.g., watching and buying), they will act on other items similarly.

Collaborative Filtering

- Coined by Goldberg et al. in Tapestry (1992): "people collaborate to help one another perform filtering by ..."

- **Assumption**
  - If two users act on n items similarly (e.g., watching and buying), they will act on other items similarly.

- **Two main phases**

  (1) Offline model-building

  (2) On-demand recommendation

Collaborative Filtering

- Coined by Goldberg et al. in Tapestry (1992): "people collaborate to help one another perform filtering by ..."
- **Assumption**
  - If two users act on n items similarly (e.g., watching and buying), they will act on other items similarly.
- **Two main phases**
  (1) Offline model-building
  (2) On-demand recommendation
- **Challenges**
  - Deal with highly sparse data
  - Scale with the increasing numbers of users and items
  - Make recommendations in real time

Real-Time Collaborative Filtering

- **Top N item recommendation**

  Given a target user u, to recommend a list of items $c_1, \ldots, c_m$ such that

  $$\mathcal{A}(u, c_1) \geq ... \geq \mathcal{A}(u, c_m)$$

  where $\mathcal{A}(u, c_i)$ $(i = 1, \ldots, m)$ are the highest prediction scores of how much u would be interested in c_i.

Real-Time Collaborative Filtering

- **Top N item recommendation**

  Given a target user u, to recommend a list of items $c_1, \ldots, c_m$ such that

  $$\mathcal{A}(u, c_1) \geq ... \geq \mathcal{A}(u, c_m)$$

  where $\mathcal{A}(u, c_i)$ $(i = 1, \ldots, m)$ are the highest prediction scores of how much u would be interested in c_i.

- **Some questions**

  – How to conduct pair-wise comparisons efficiently?
  e.g., user-user/item-item

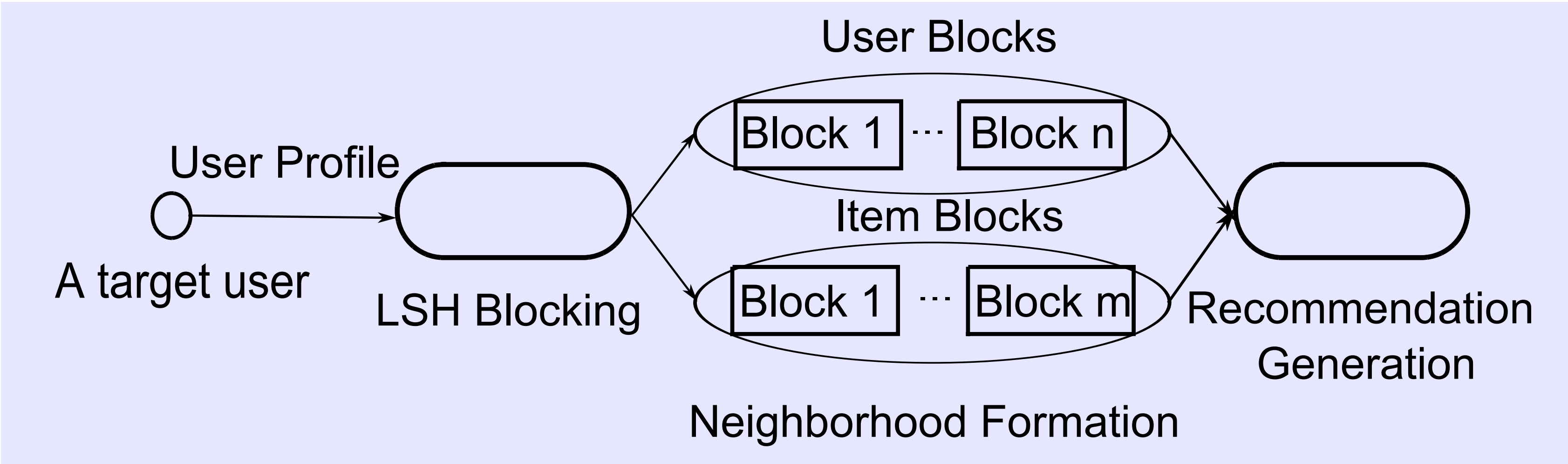
  – How to capture new updates quickly?
  e.g. latest updates in social media

Overview of the Proposed Approach

- **Key components**
  - LSH blocking
  - Neighbourhood formation
  - Recommendation generation

Overview of the Proposed Approach

- **Key components**
  - LSH blocking
  - Neighbourhood formation
  - Recommendation generation



LSH Blocking

- **Construct blocks** based on Cosine similarities
  - User blocks
  - Item blocks

LSH Blocking

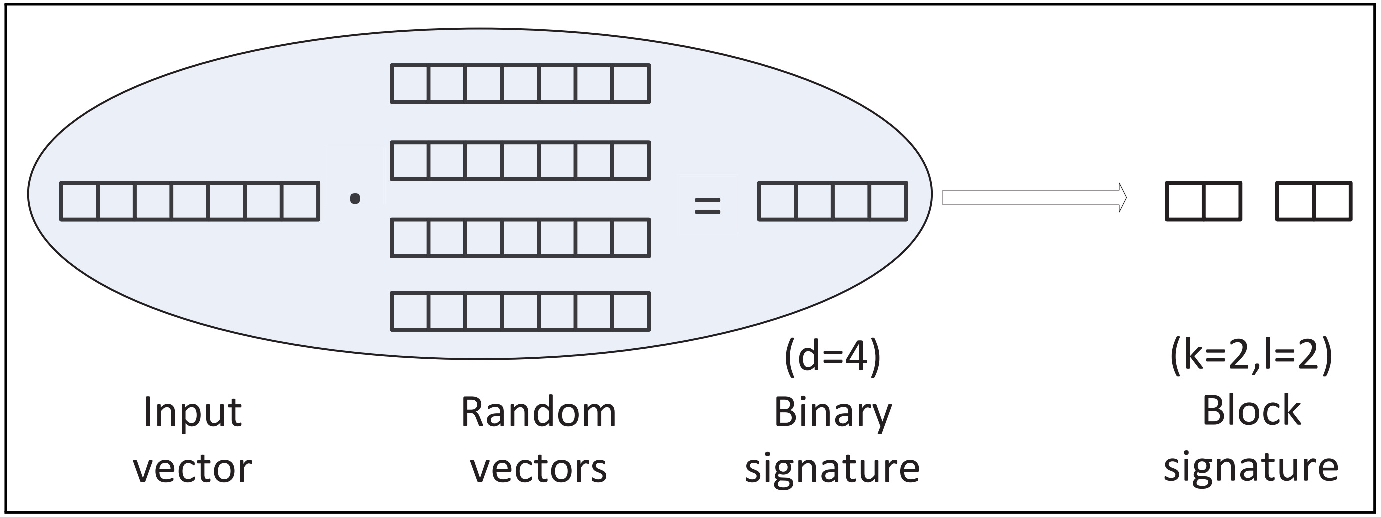
- **Construct blocks** based on Cosine similarities
  - User blocks
  - Item blocks

- **Use two LSH families** to approximate Cosine similarities

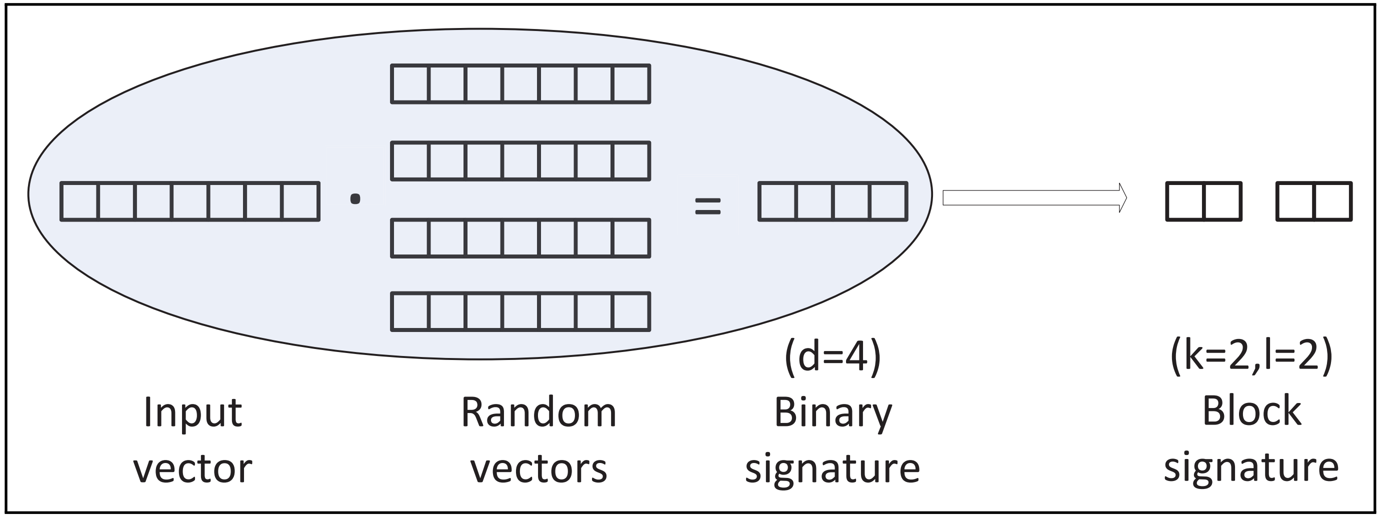
  (1) Random hyperplane projection

  (2) Random bit sampling

LSH Blocking
– Random Hyperplane Projection

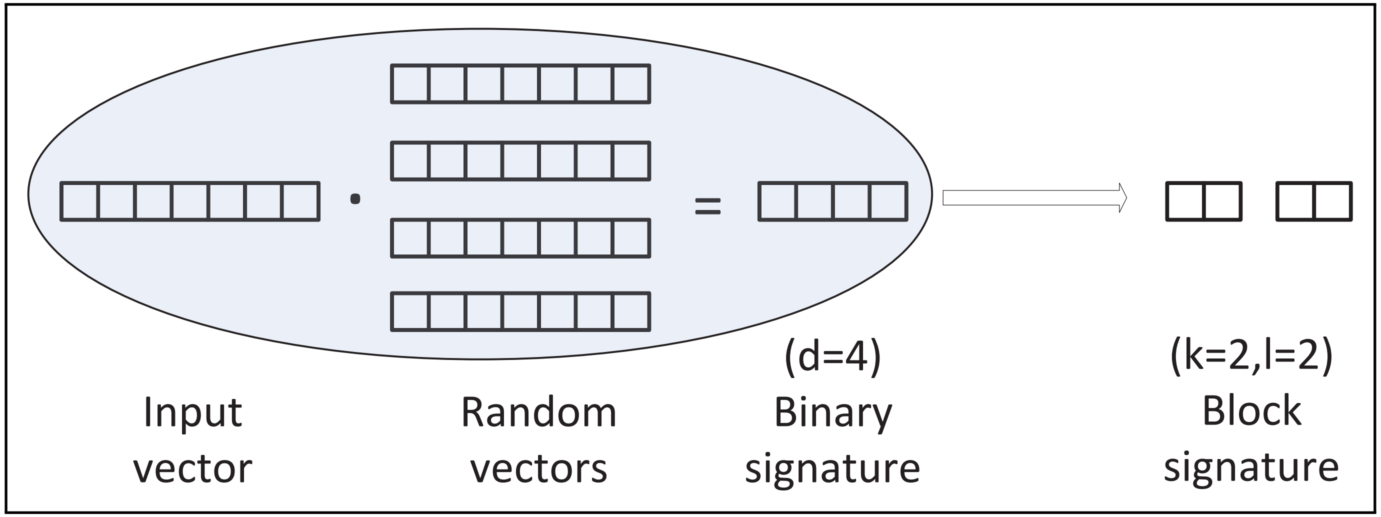


LSH Blocking
– Random Hyperplane Projection



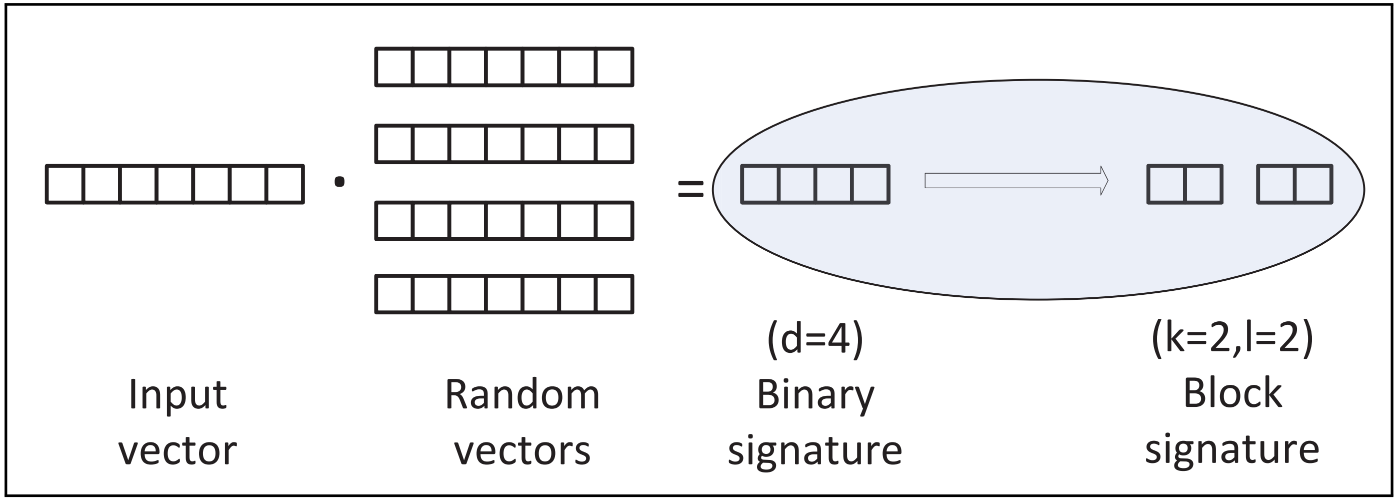
- A n-dimensional input vector is mapped to a d-bit binary signature using random vectors, usually $d \ll n$.

LSH Blocking
– Random Hyperplane Projection

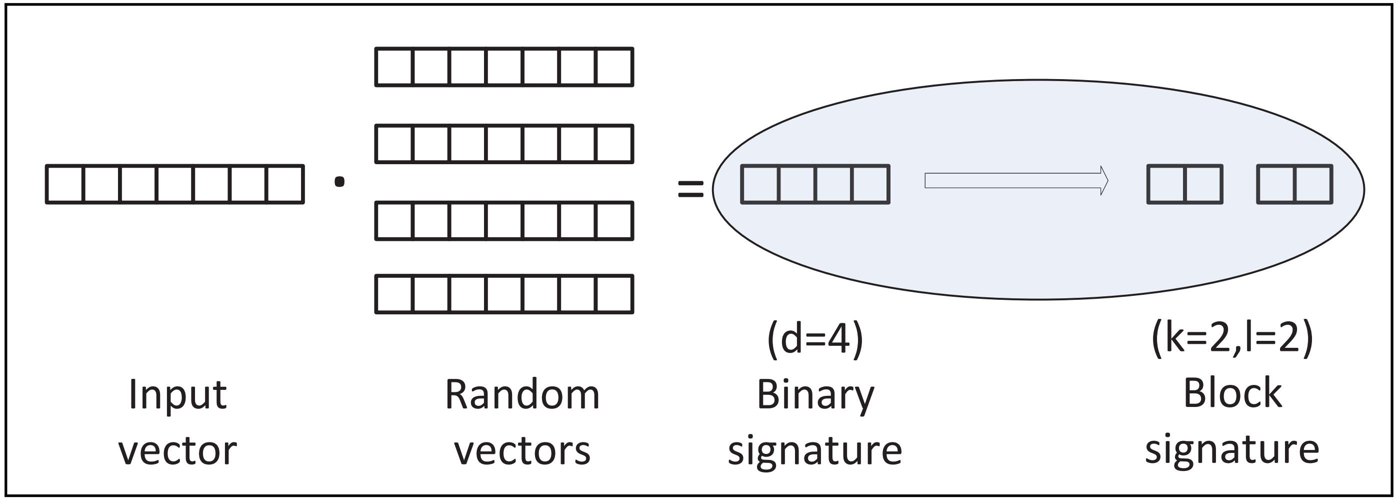


- A n-dimensional input vector is mapped to a d-bit binary signature using random vectors, usually $d \ll n$.
- The more random vectors we use, the more accurate the Cosine similarity between two input vectors is.

LSH Blocking
– Random Bit Sampling

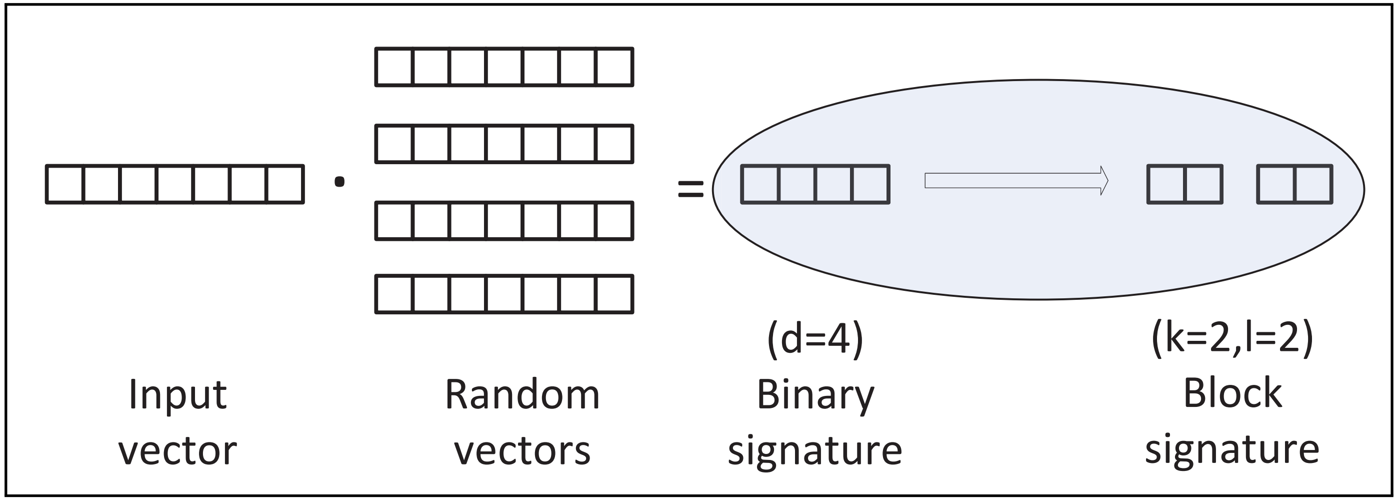


LSH Blocking
– Random Bit Sampling



- Use the Hamming distance to measure the similarity of two binary signatures

LSH Blocking
– Random Bit Sampling



- Use the Hamming distance to measure the similarity of two binary signatures
- Use random bit sampling to approximate the Hamming distance over $\{0,1\}^d$
  - Select random bits from the binary signatures
  - Amplify the collision probability using AND/OR constructions

Neighborhood Formation

- Use user and item blocks to identify the neighbor users/items
  - Neighbor users: in the same user blocks as a user
  - Neighbor items: in the same item blocks as an item

Neighborhood Formation

- Use user and item blocks to identify the neighbor users/items
  - Neighbor users: in the same user blocks as a user
  - Neighbor items: in the same item blocks as an item

- But, user/item blocks could still be large ...

Neighborhood Formation

- Use user and item blocks to identify the neighbor users/items
  - Neighbor users: in the same user blocks as a user
  - Neighbor items: in the same item blocks as an item

- But, user/item blocks could still be large ...

- how to efficiently make the top N recommendations for a target user based on neighbor users/items?

Real-time Recommendation Generation

- **Two approaches**
  - User-based recommendation
  - Item-based recommendation

Real-time Recommendation Generation – User-based Recommendation

- **Rank/select neighbor users**
  - Count collision numbers of neighbour users in user blocks with the target user
  - Set a threshold on the collision numbers to select neighbor users

Real-time Recommendation Generation – User-based Recommendation

- **Rank/select neighbor users**
  - Count collision numbers of neighbour users in user blocks with the target user
  - Set a threshold on the collision numbers to select neighbor users

- **Calculate prediction scores**
  - Find candidate items from the items of selected neighbor users
  - Calculate the similarities between the target user and neighbor users who have a candidate item:

$$\mathcal{A}_u(u_i, c_x) = \sum_{u_j \in \mathbf{N}_{u_i} \bigcap U_{c_x}} \frac{1}{\sqrt{|\mathbf{N}_{u_i} \bigcap U_{c_x}|}} \cdot cosine(\mathbf{u_i}, \mathbf{u_j})$$

Real-time Recommendation Generation – User-based Recommendation

- **Rank/select neighbor users**
  - Count collision numbers of neighbour users in user blocks with the target user
  - Set a threshold on the collision numbers to select neighbor users
- **Calculate prediction scores**
  - Find candidate items from the items of selected neighbor users
  - Calculate the similarities between the target user and neighbor users who have a candidate item:

$$\mathcal{A}_u(u_i, c_x) = \sum_{u_j \in \mathbf{N}_{u_i} \bigcap U_{cx}} \frac{1}{\sqrt{|\mathbf{N}_{u_i} \bigcap U_{cx}|}} \cdot cosine(\mathbf{u_i}, \mathbf{u_j})$$

- **Generate recommendations**
  - The top N items with high prediction scores

Real-time Recommendation Generation – Item-based Recommendation

- **Rank/select neighbor items**
  - Count collision numbers of neighbour items in item blocks with each item of the target user
  - Set a threshold on the collision numbers to select neighbor items

Real-time Recommendation Generation – Item-based Recommendation

- **Rank/select neighbor items**
  - Count collision numbers of neighbour items in item blocks with each item of the target user
  - Set a threshold on the collision numbers to select neighbor items

- **Calculate prediction scores**
  - Find candidate items, i.e., all selected neighbour items
  - Calculate the similarities between each item of the target user and a candidate item:

$$\mathcal{A}_c(u_i, c_x) = \sum_{c_j \in C_{u_i}} \frac{1}{\sqrt{|C_{u_i}|}} \cdot cosine(\mathbf{c_j}, \mathbf{c_x})$$

Real-time Recommendation Generation – Item-based Recommendation

- **Rank/select neighbor items**
  - Count collision numbers of neighbour items in item blocks with each item of the target user
  - Set a threshold on the collision numbers to select neighbor items

- **Calculate prediction scores**
  - Find candidate items, i.e., all selected neighbour items
  - Calculate the similarities between each item of the target user and a candidate item:

$$\mathcal{A}_c(u_i, c_x) = \sum_{c_j \in C_{u_i}} \frac{1}{\sqrt{|C_{u_i}|}} \cdot cosine(\mathbf{c_j}, \mathbf{c_x})$$

- **Generate recommendations**
  - The top N items with high prediction scores

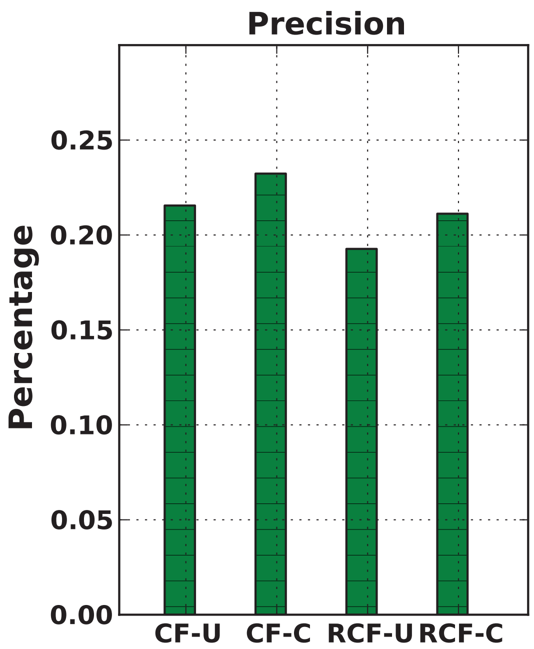
Experimental Setup

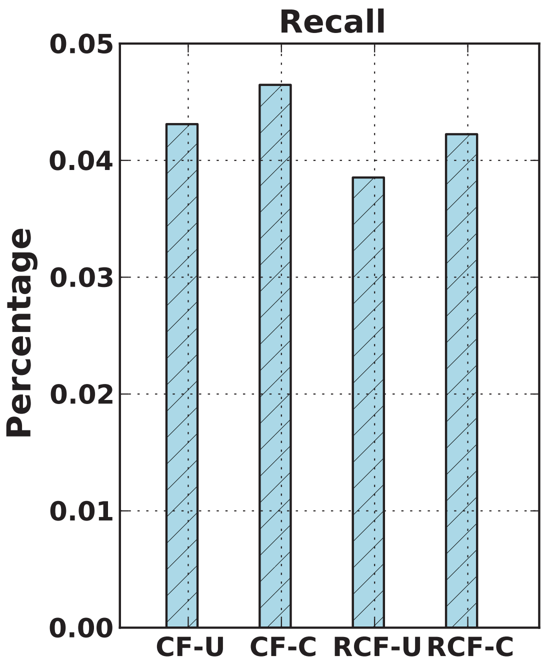
- **Experiment**
  - Topic recommendation (i.e., recommend topics to users in a social media community)
- **Data set**
  - Crawled from Twitter.com
  - Selects the keywords that are at least used by 5 users as topics, and the users who have used at least 5 topics
  - Contains 2320 users, 3319 topics, and 1,214,604 tweets
  - Split into 90% training (2088 users) and 10% test (232 users)
- **Evaluation metrics**
  - Top N=10 Precision & Recall
  - Average Recommendation Time

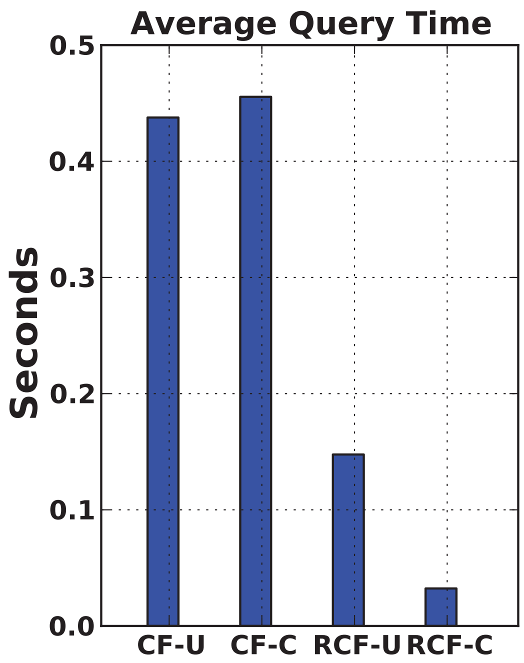
Experimental Results

– **Compared approaches**

- CF-U & CF-C: Traditional user & item based CF
- RCF-U & RCF-C: Real-time user & item based CF







Conclusions

- We have studied a real-time recommender system
  - LSH Blocking
  - Neighborhood formation
  - Recommendation generation
- We have used two LSH families to approximate the similarities between items/users
  - Random hyperplane projection
  - Random bit sampling
- We have conducted experiments on a Twitter dataset
- As future work, the temporal aspects of items and users can be future considered