INCREMENTAL KNN

Author: Wei Ma

Supervisor: Rhicheek Patra

Content

- KNN(K Nearest Neighbors)
- How to improve KNN
  - Sample Potential candidates
  - Distributes tasks to the clients
- DeKNN – Democratization KNN
  - KMeans DeKNN
  - Item DeKNN
  - Comparison with HyRec
- Experiments
- Application on Spam
- Conclusion

KNN

- Lazy learning

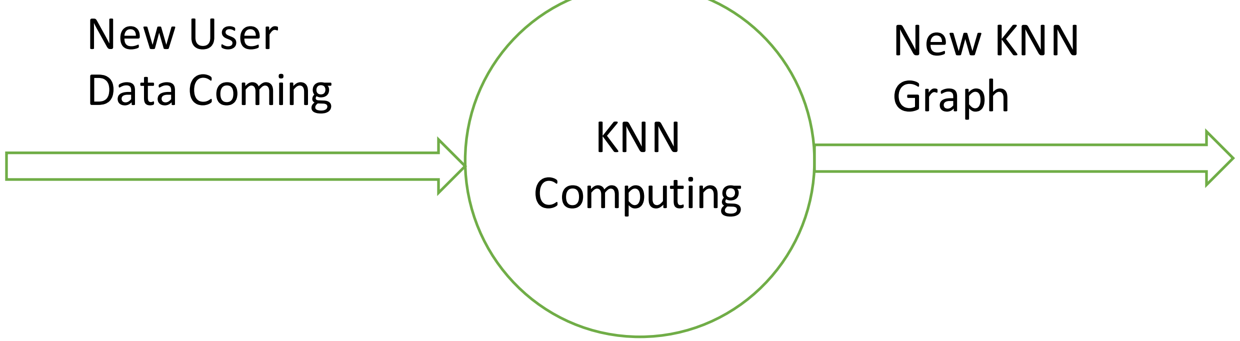
  - Defers data processing until it receives a request to classify unlabeled data

  - Replies to a request for information by combining its stored training data

  - Discards the constructed answer and any intermediate results

Problems with KNN in practice

- KNN is an efficient method for the Recommendation System.



How To Solve It

- Using other methods to approximate KNN graph. Don’t compute KNN by force.

Sample Potential Candidates!

Let the clients undertake some tasks.

- Chain Rule: Sample candidates from the neighbors
  - If A is highly similar with B and B is highly similar with C, A is possible to be highly similar with C.

Chain Rule is EFFICIENT but not ENOUGH.

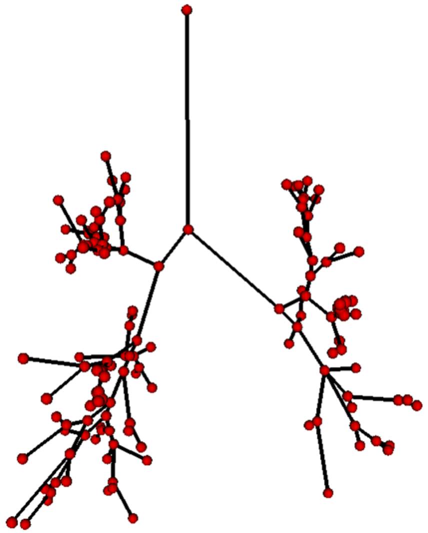
- Why
  - . In most cases, it can work well
  - . Special Case.

    Three users: A (i1,i2,i3) B (i1,i2,i4,i5) C(i4,i5,i6).

    A is not similar with C at all!

- Using other methods to improve the quality of the potential candidates.
  - -Cluster the data
    - . Kmeans
    - . Item based
  - -Randomly choose some users from the whole data to avoid the local optimization.

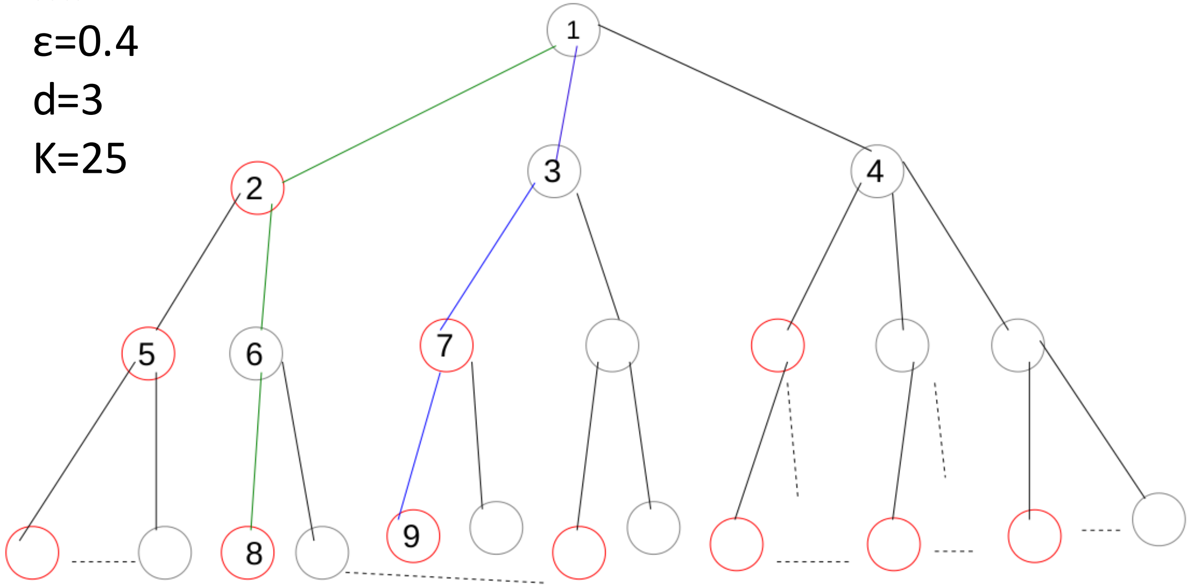
Candidates from Neighbors - based on Chain Rule



Explore and Exploit the graph

Explore and Exploit the graph

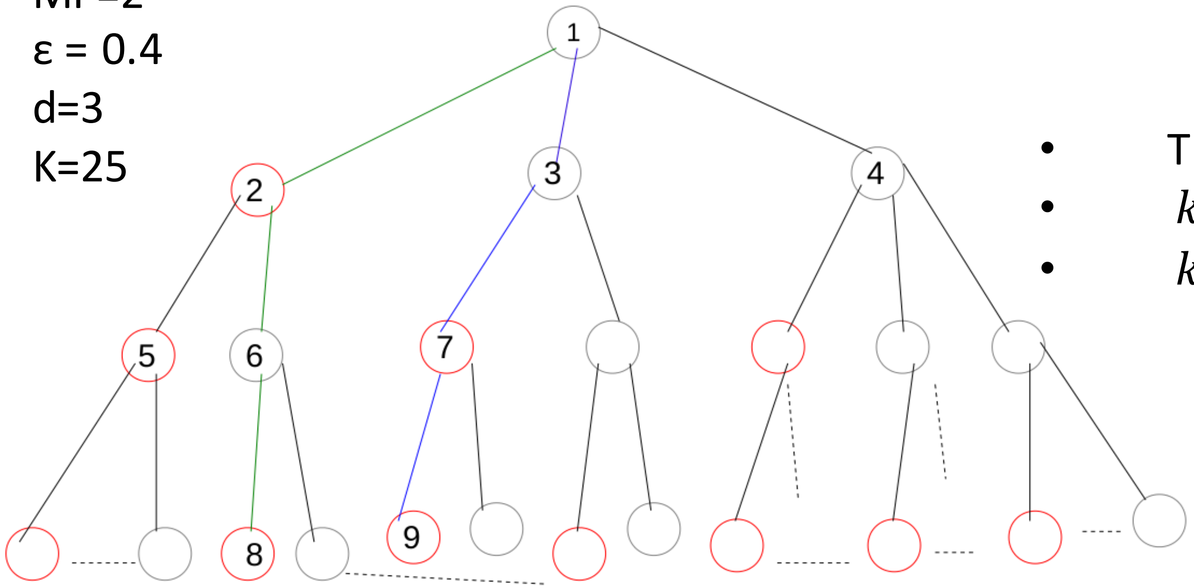
- MP=2
- ε=0.4
- d=3
- K=25



- MP: mutable path. How many paths we search in the graph. Its up bound is a$*log_2$K. a is usally 1.
- ε: the probability to choose not the most similar neighbor
- K is the parameter of KNN
- d is the search depth and is constant.

Mutable Path

- MP=2
- ε = 0.4
- d=3
- K=25



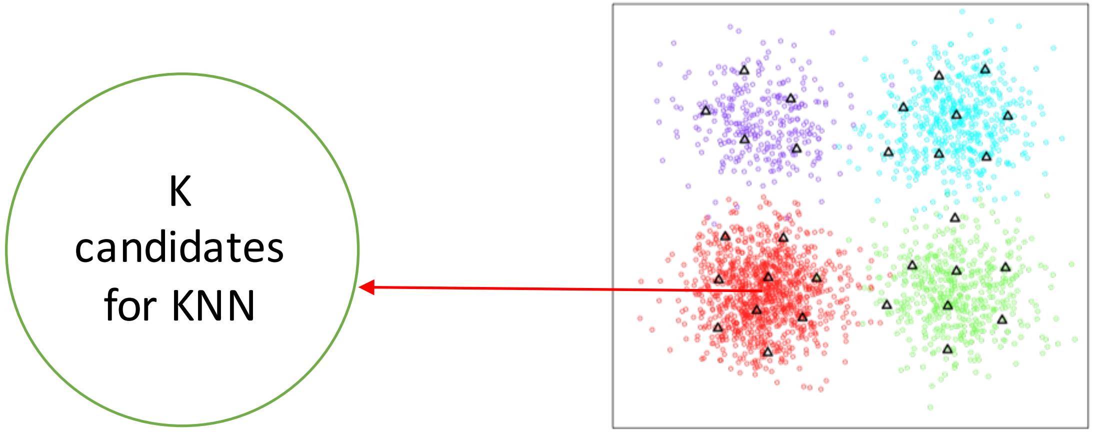
$$|knn_{pre} \cap knn_{after}| < \theta$$

- Theta is an predefined threshold according to K.
- knn_{pre} *is the top k neighbors of node* 1 *before updating*.
- knn_{after} is the top k neighbors of node 1 after updaing.

$$The\ size\ of\ candidates\ form\ neighbors\ is\ not\ more\ than\ K * a * log_2 K$$

DeKNN – Democratization KNN

- KMeans DeKNN
  - Using Online KMeans to classify the users.
  - Choose K candidates from the clusters.



Online Kmeans – Update Clusters

Input: A new user profile at time t,noted as U^t and K-Means' clusters, noted as C

Output: A new k-Means center

$C_j \leftarrow$ **find the most similar cluster with the** U^t

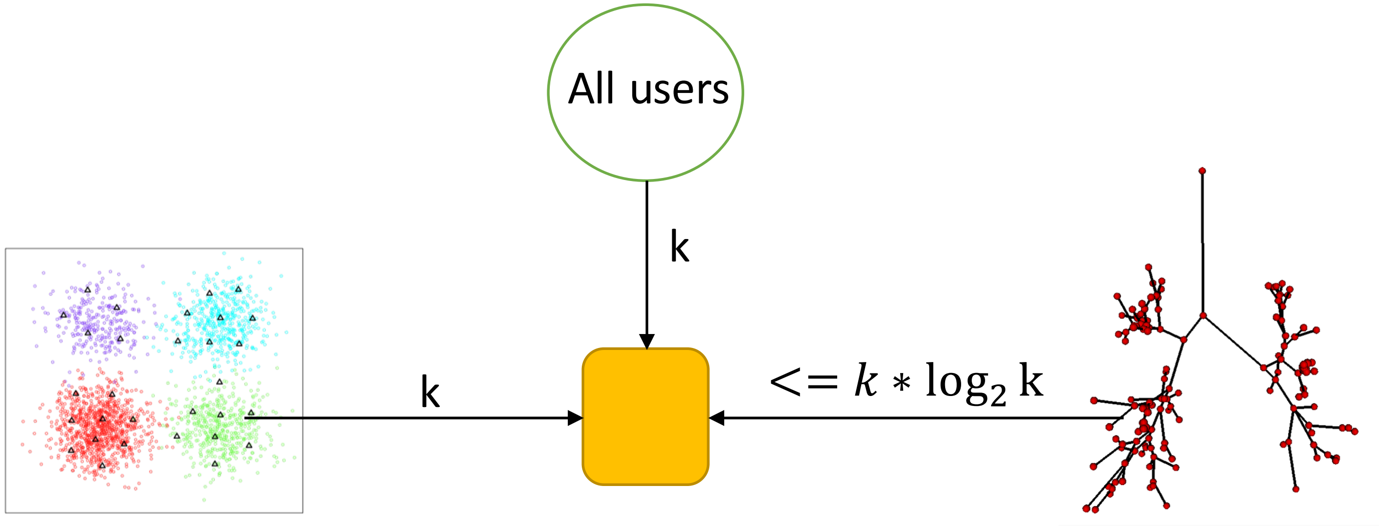
learning rate, $r = \frac{1}{|C_j|+1}$

update the cluster's center, $C_j = C_j + (U^t - C_j) \times r$

KM-DeKNN (Kmeans DeKNN)

- Choosing Candidates. Its size up bound $2\mathrm{k} + \mathrm{k} * \log_2 k$ and it low bound is $3\mathrm{k}$

- sample k candidates from the whole users.
- sample at least k and at most $k * \log_2 \mathrm{k}$ candidates from the neighbors.
- sample k candidates from the clusters.



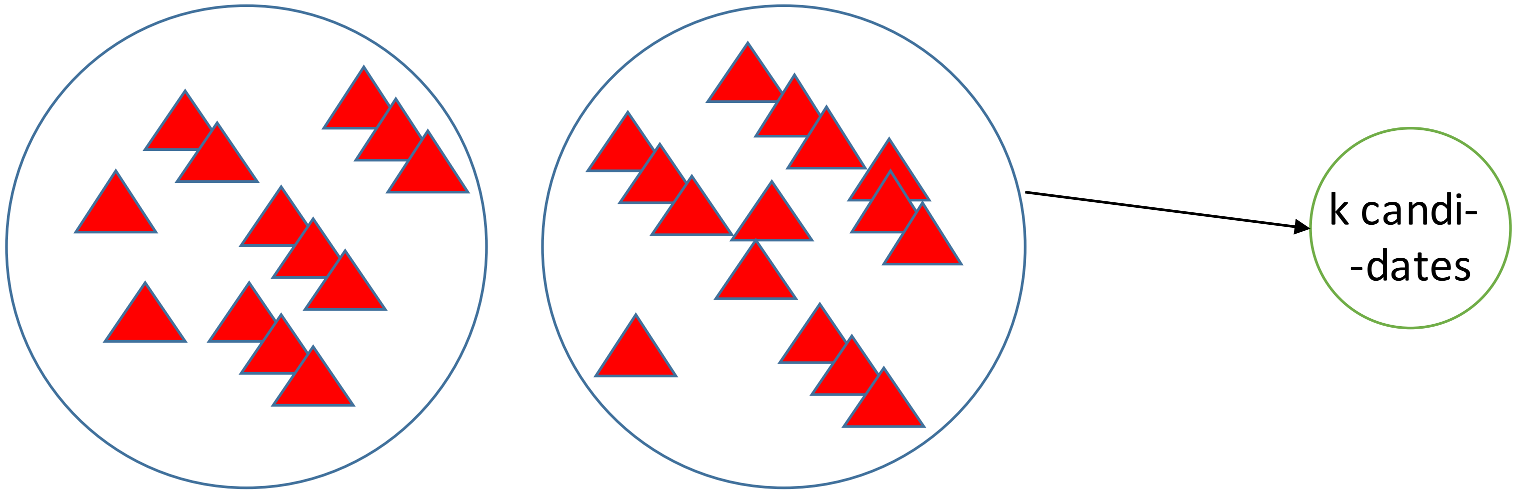
Problems with KM-DeKNN

- The number of the clusters are rough to be decided.
- The number of the clusters are fixed and immutable. Cannot find the new structures of the users.
- With the increase of the users, each cluster will contain more and more users and become huge and stubborn.
- How to solve it?

Item-based DeKNN(It-DeKNN)

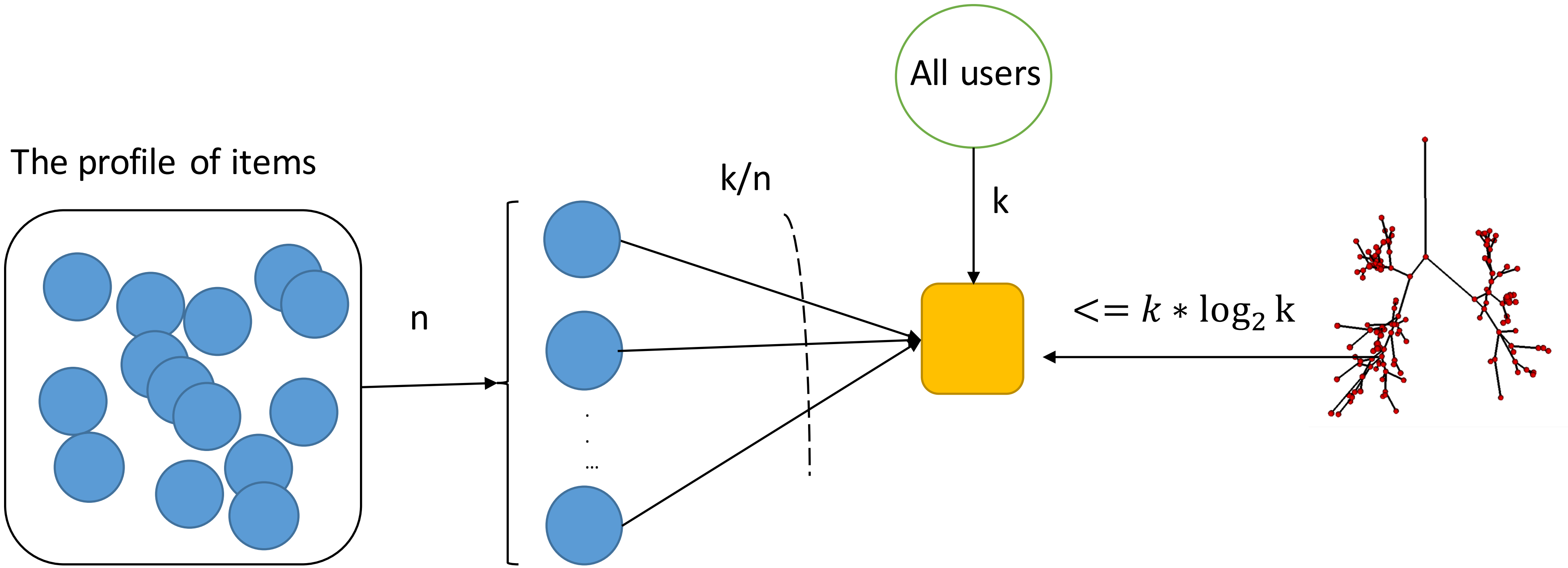
It-DeKNN

- The positively similar users must buy the same types of the products.
- Each item maintains its users list who bought it in the past.
- Difference with Item DeKNN and Kmeans KNN.



A circle is the profile of the item.
A triangle is a user

Item-Based DeKNN

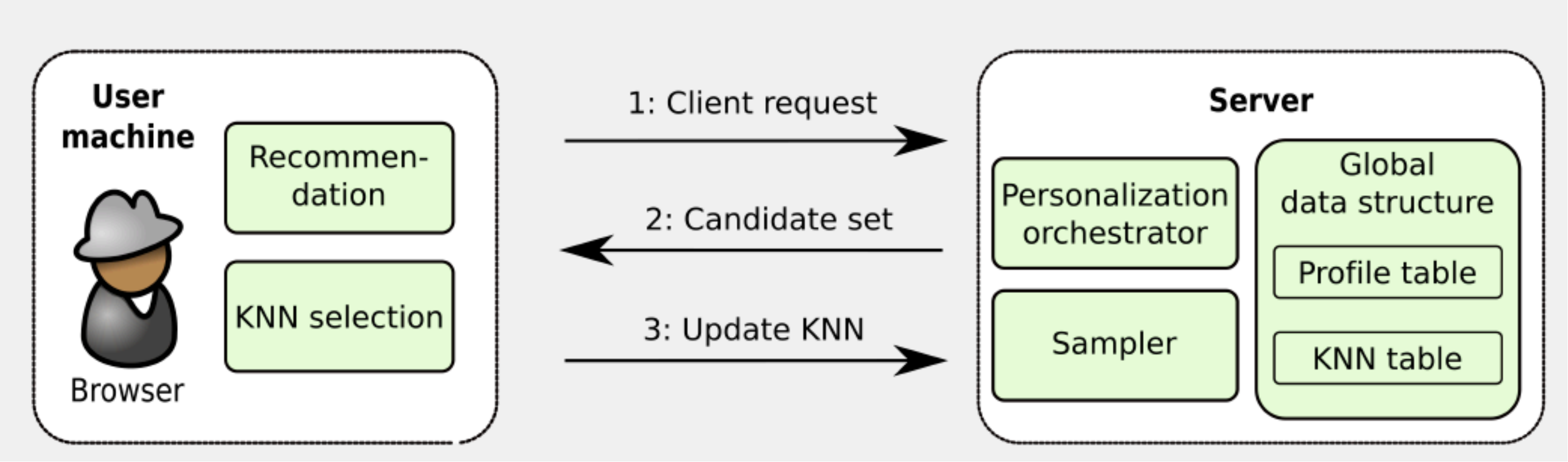


Distribute Tasks to the Clients

- Computing the similarity between the users needs much time.

  Send the candidates selected on the server to the clients.

  The client will compute the similarity and select top K neighbors.



Experiments

- Compare DeKNN with HyRec[1]
  - HyRec's sample candidate size $k^2 + 2k$
  - HyRec will sample a candidate set for a user u at time t by aggregating three sets.
    - the current approximation of u's KNN (k)
    - the current neighbors of all the u's KNN neighbors (k^2)
    - k random users

Boutet, Antoine, Davide Frey, Rachid Guerraoui, Anne-Marie Kermarrec, and Rhicheek Patra. "Hyrec: Leveraging browsers for scalable recommenders."

Experiment

- Goal
  - DeKNNs should converge eventually.
  - DeKNNs should be faster than HyRec
  - DeKNNs has smaller candidate set than HyRec does.
    - It will lead to DeKNNs need less storage than HyRec in the client.
- Dataset
- Performance
- Application : ItDeKNN Spam

Parameters

- K = 25
- d=5
- $\varepsilon = 0.5$
- $\theta = 6$
- The up bound of MP is $k * \log_2 \mathrm{k}$.

Predication

80% data for train and 20% data for test

KNN-Recall

$$recall = \sum_{u \in U} \left(\frac{|\widehat{knn_u} \cap knn_u|}{k} \right) / |U|$$

- U is the set of the users.
- $\widehat{knn}_u$ *is the approximating top k neighobrs .*
- knn_u *is the true top k neighbors*

Predicating Rate

$$\tilde{r}(u,i) = \bar{r}(u) + C_0 \sum_{v \in N_k(u,i)} sim(u,v)\left(r(v,i) - \bar{r}(v)\right)$$

- $\tilde{r}$(u,i) is the predicating rate of the user u on item i.
- $\bar{r}$ (u) is the average rating of the user u.
- sim(u,v) is the similiarity with user u and user v.
- $r(v,i)$ is the rate of the user v on item i.
- $\bar{r}$ (v) is the average rating of the user v.
- Constant C_0 is a normalization factor.

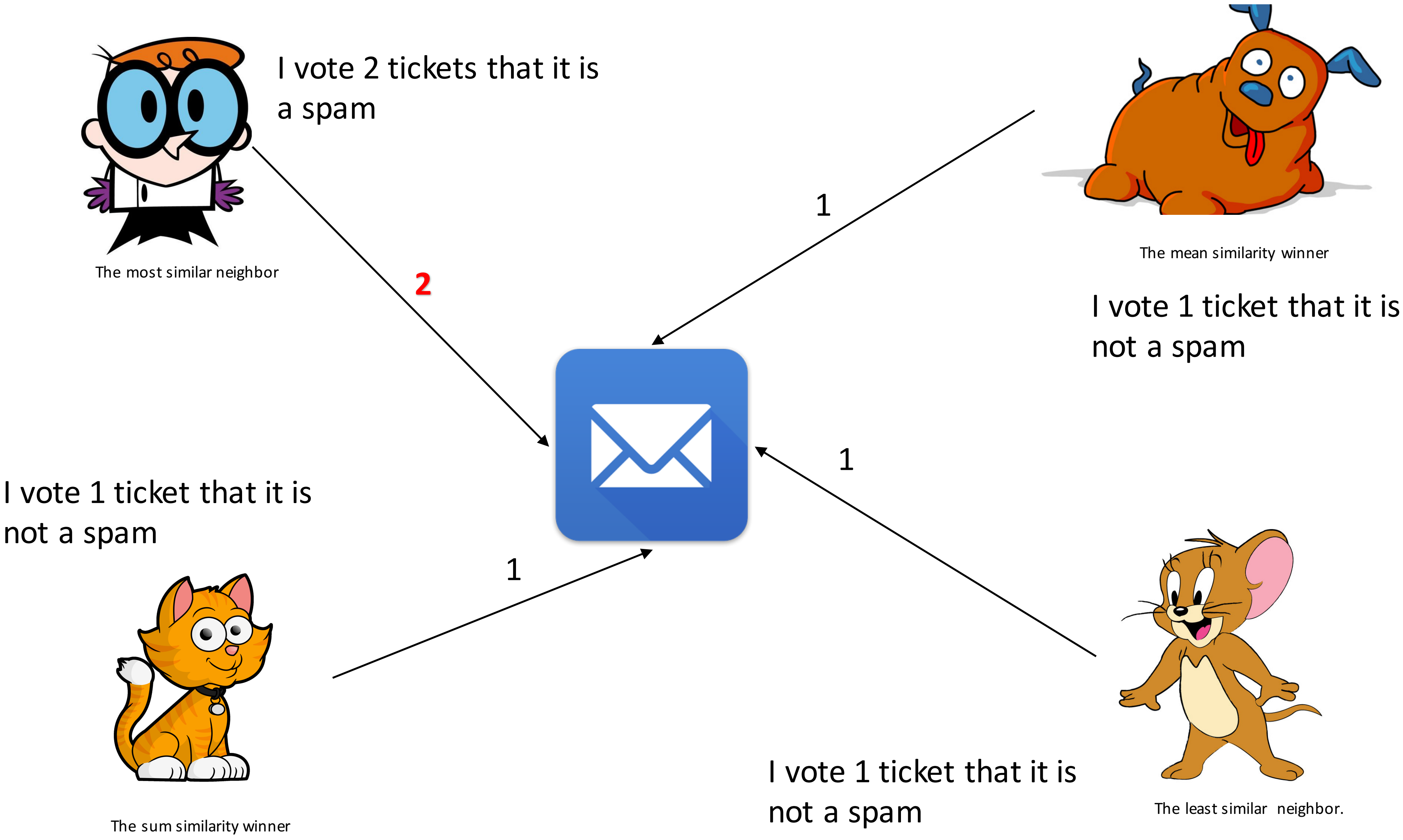
Predication Spam Recall

P\T	1	0
1	tp	fn
0	fp	tn

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

Predication Spam

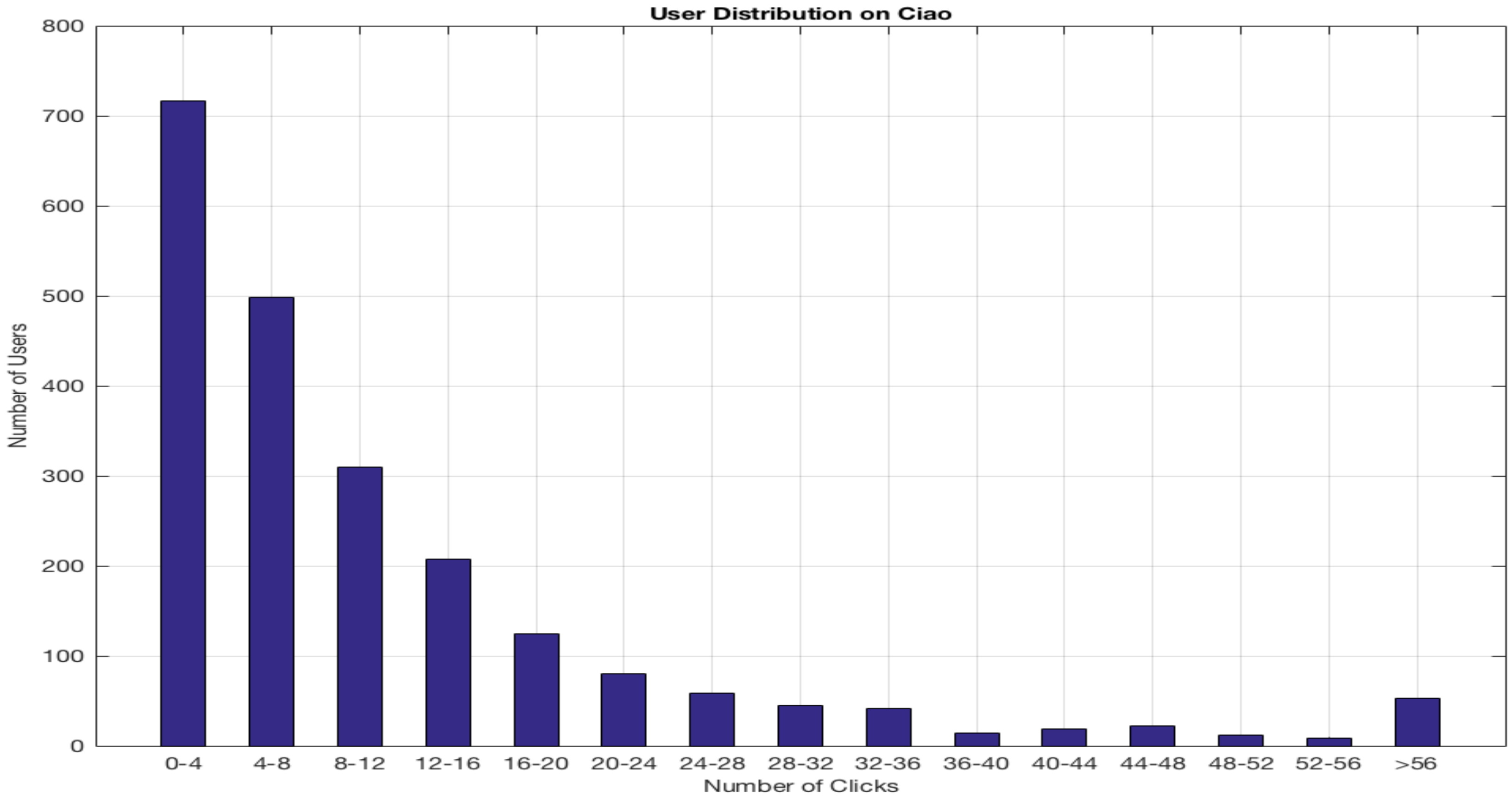
- Decide whether the email is spam or not by four methods' voting
  - The most similar neighbor have two tickets.
  - The average similarity winner has one ticket.
    - the avg similarity of spam neighbors VS the avg similarity of non-spam neighbors
  - The sum similarity winner has one ticket.
    - the sum similarity of spam neighbors VS the sum similarity of non-spam neighbors
  - The least similar neighbor has one ticket.


I vote 2 tickets that it is a spam
The most similar neighbor
2
1
The mean similarity winner
I vote 1 ticket that it is not a spam
I vote 1 ticket that it is not a spam
1
1
The sum similarity winner
I vote 1 ticket that it is not a spam
The least similar neighbor.

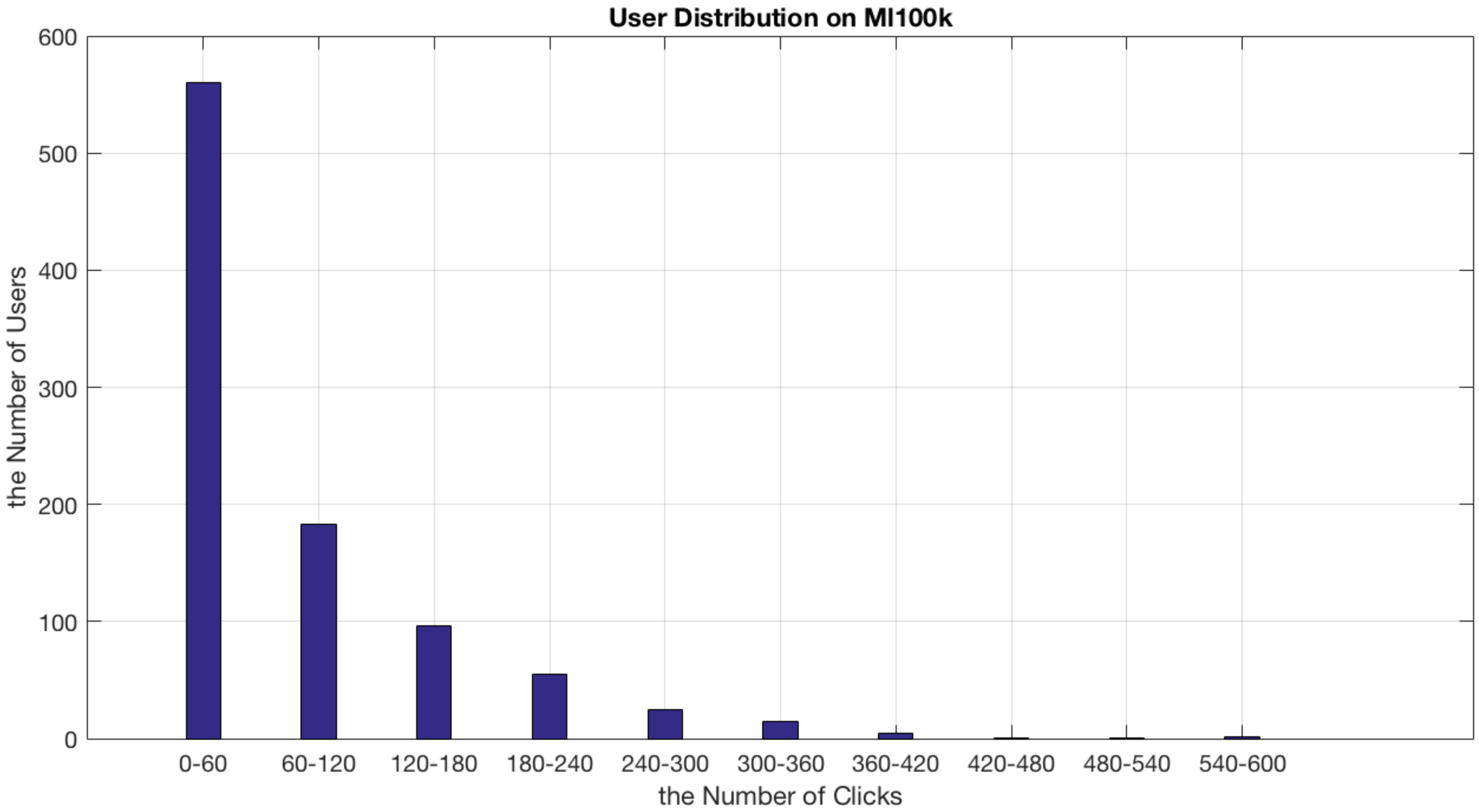
Data Description

Name	Records	Users	Rate Range(Integer)
Ciao	36065	2210	1-5
Ml-100k	100, 000	943	1-5
Ml-1M	1,000,209	6040	1-5

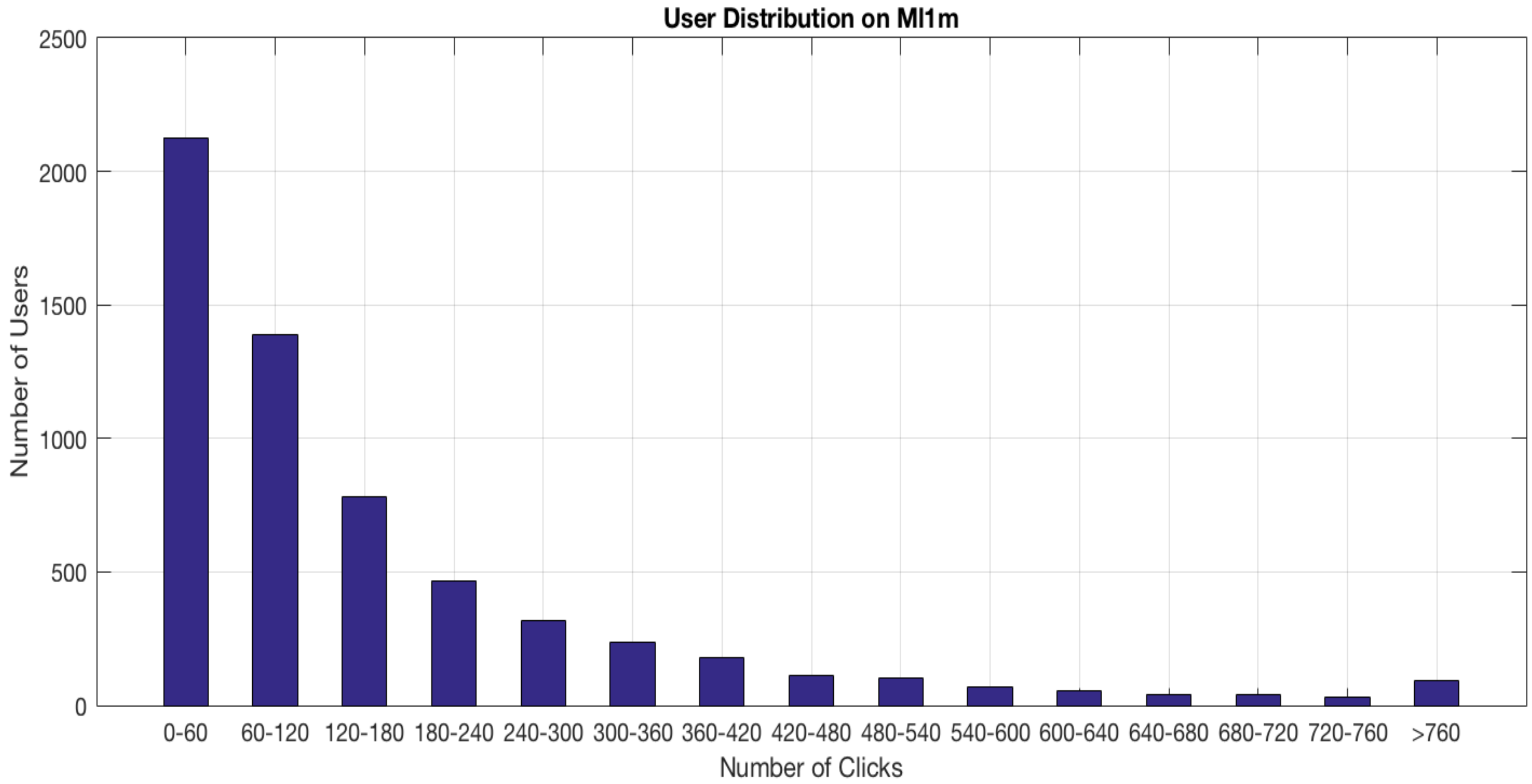
Data Description



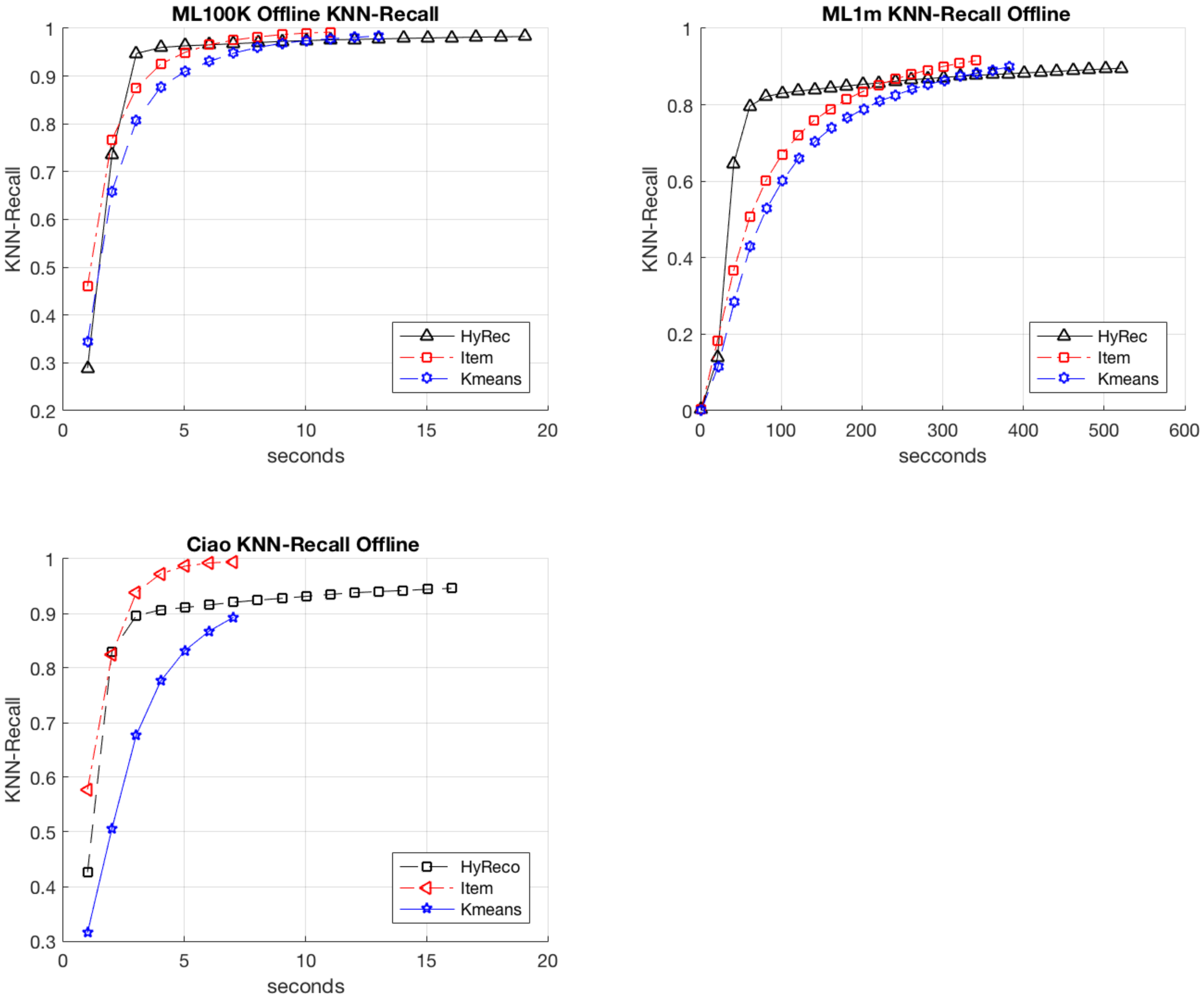
Data Description



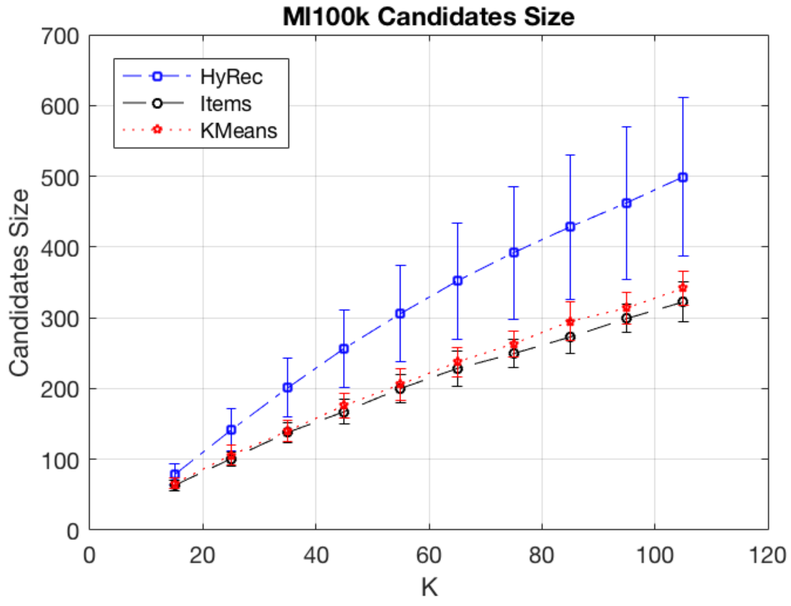
Data Description

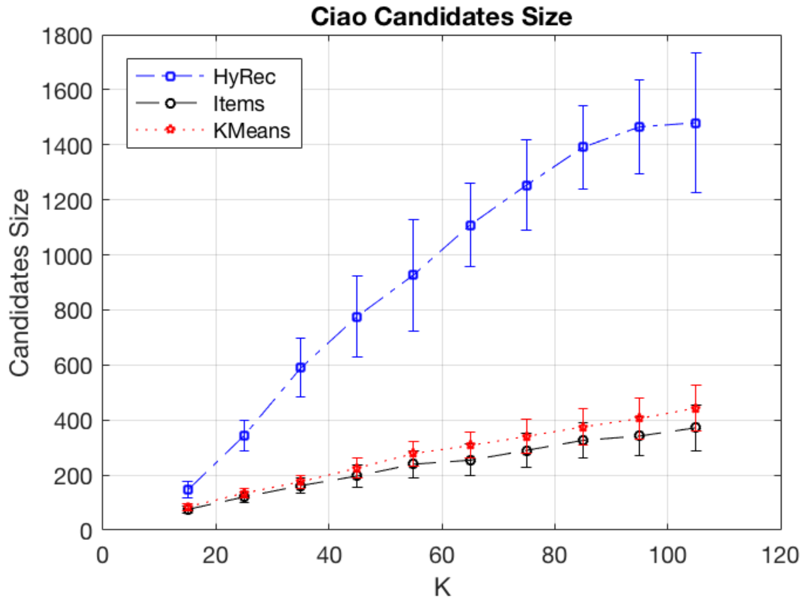


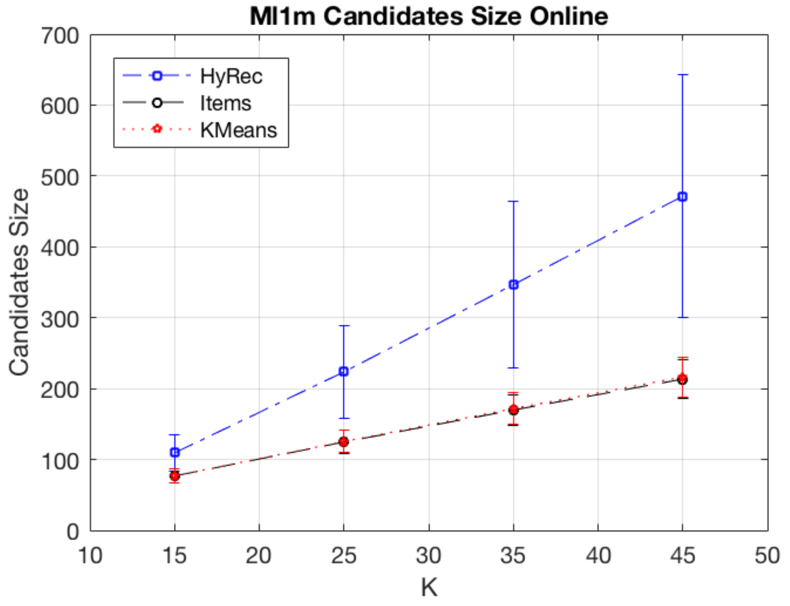
Offline KNN-Recall with Time



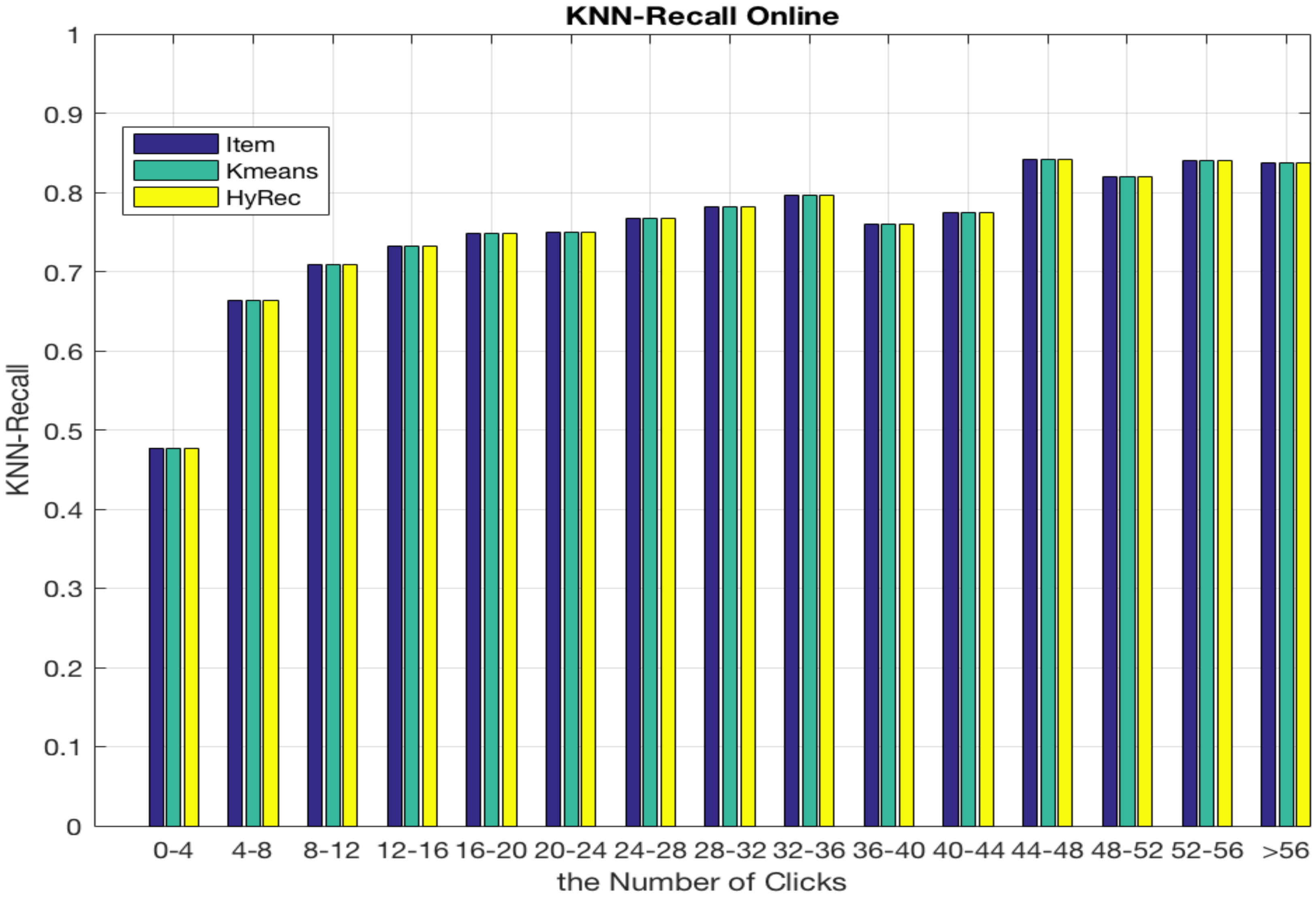
Candidate Size with different K Online



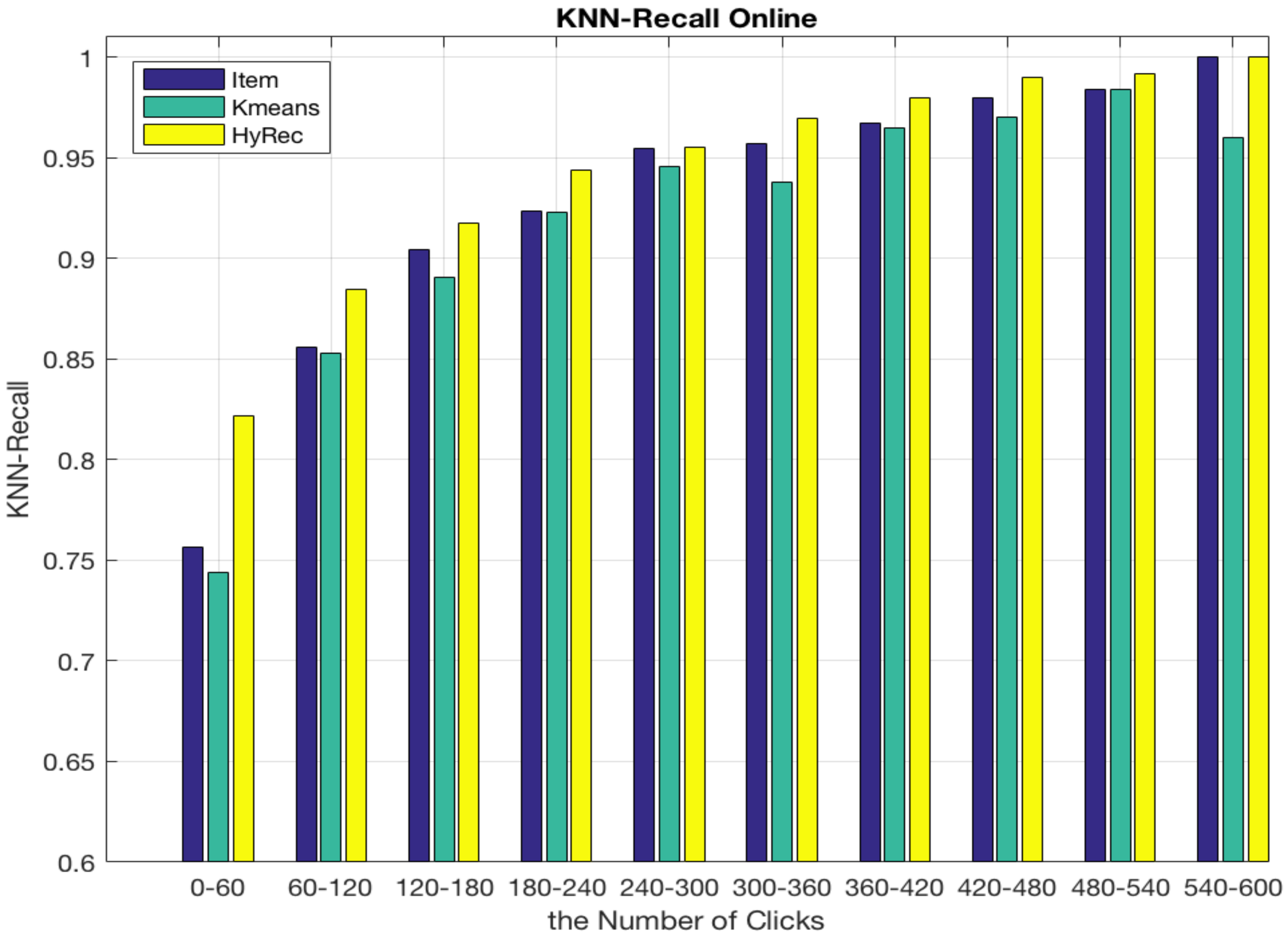




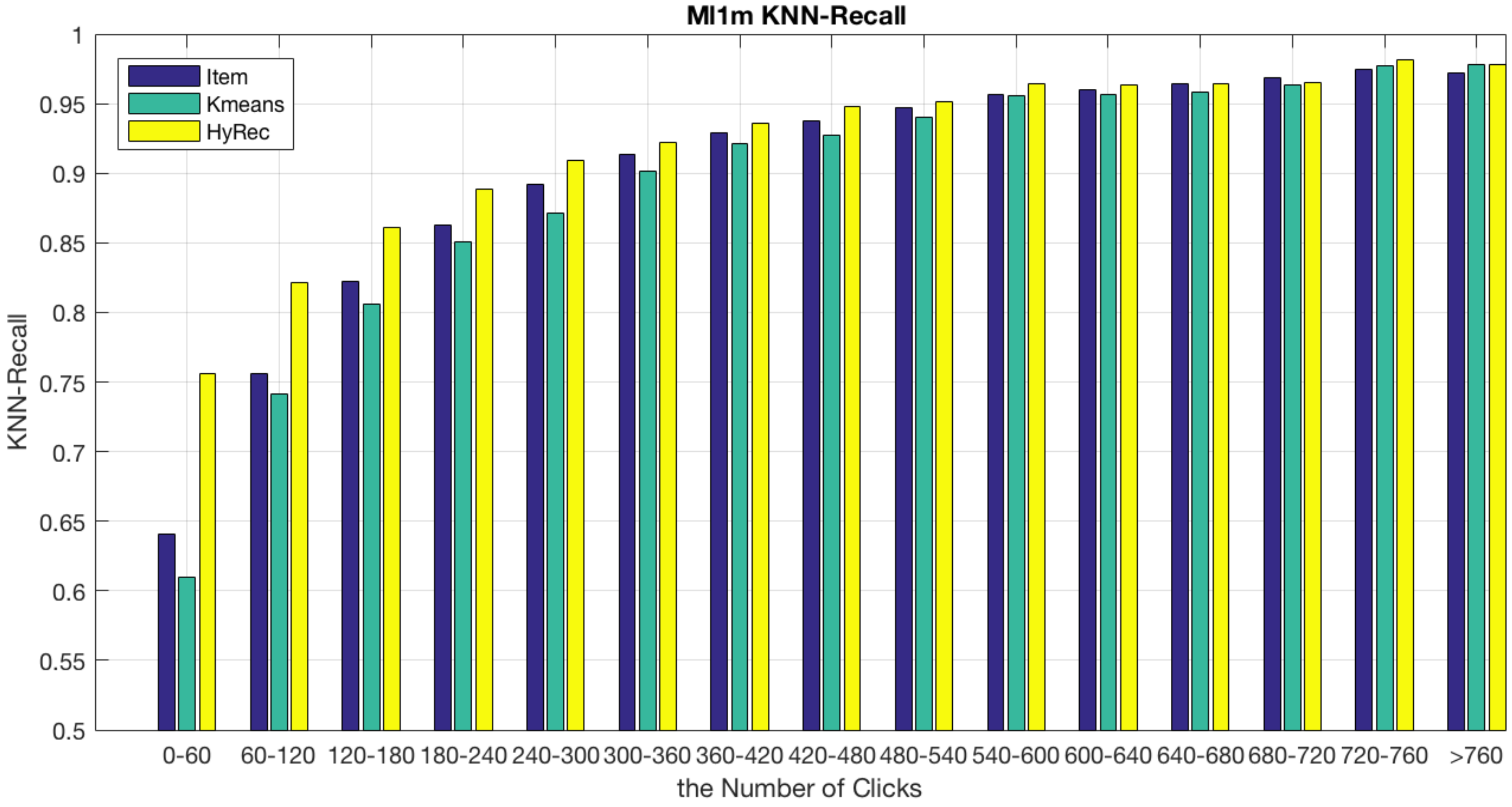
Online Recall Ciao



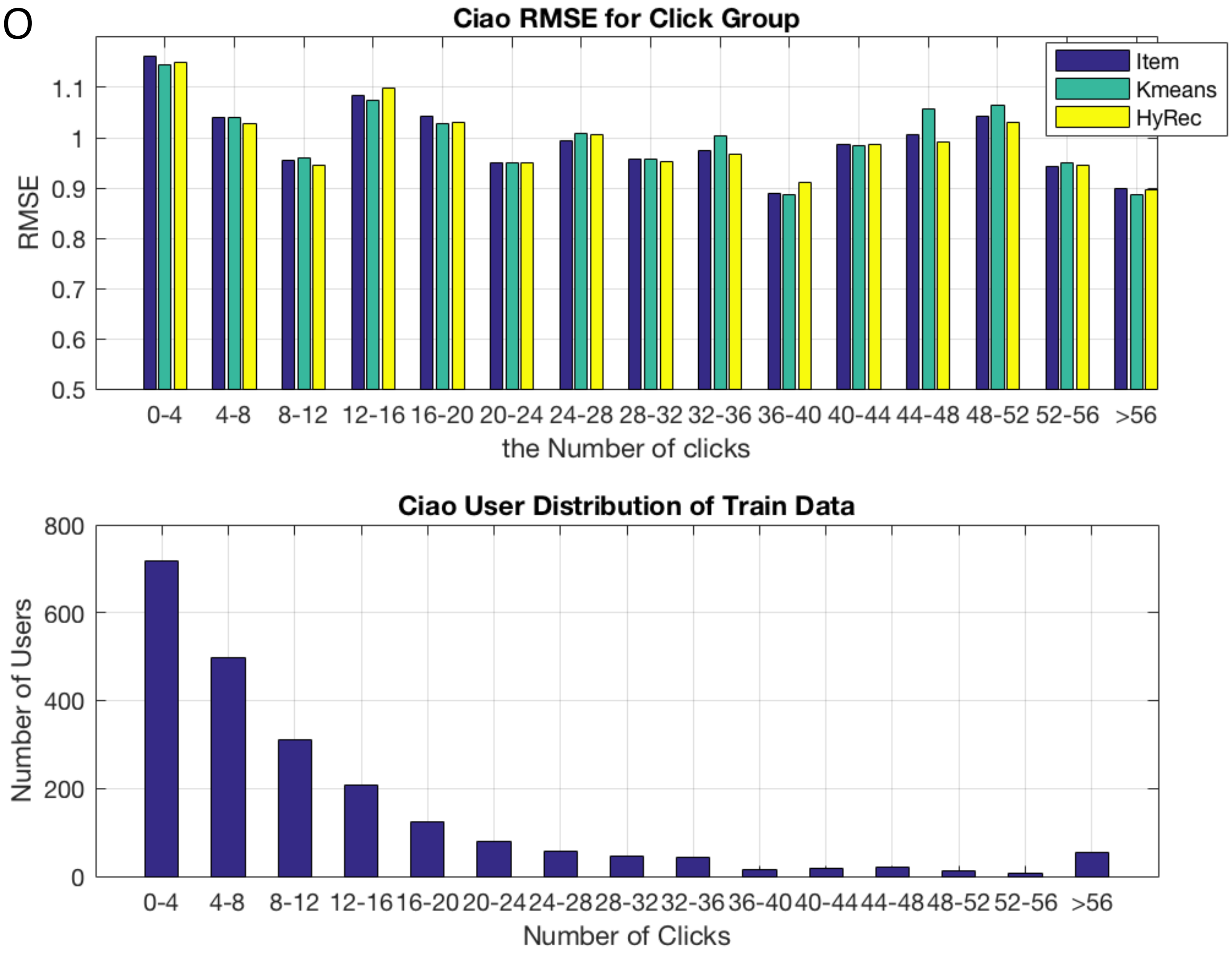
Online Recall Ml100k



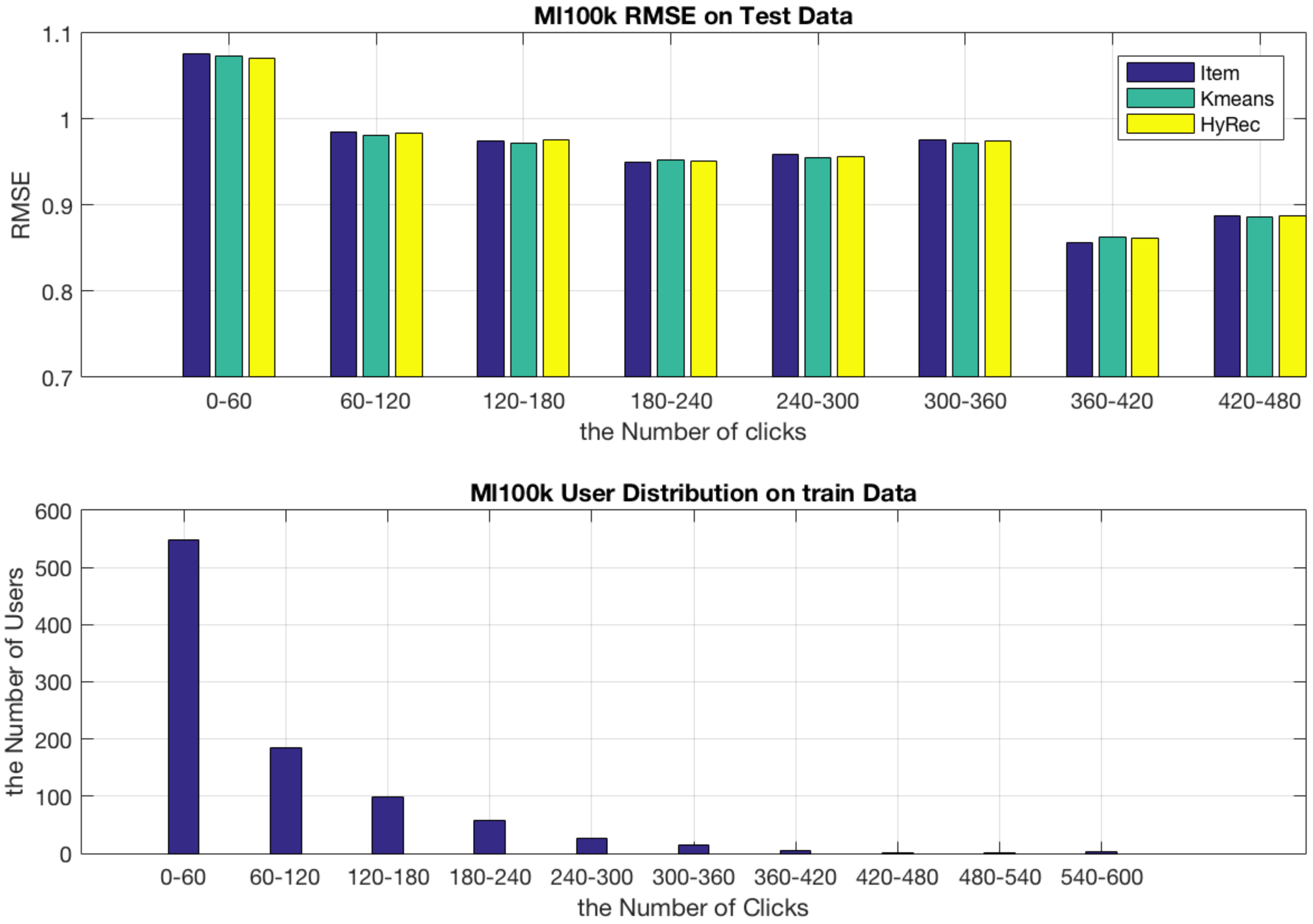
Online Recall Ml1M



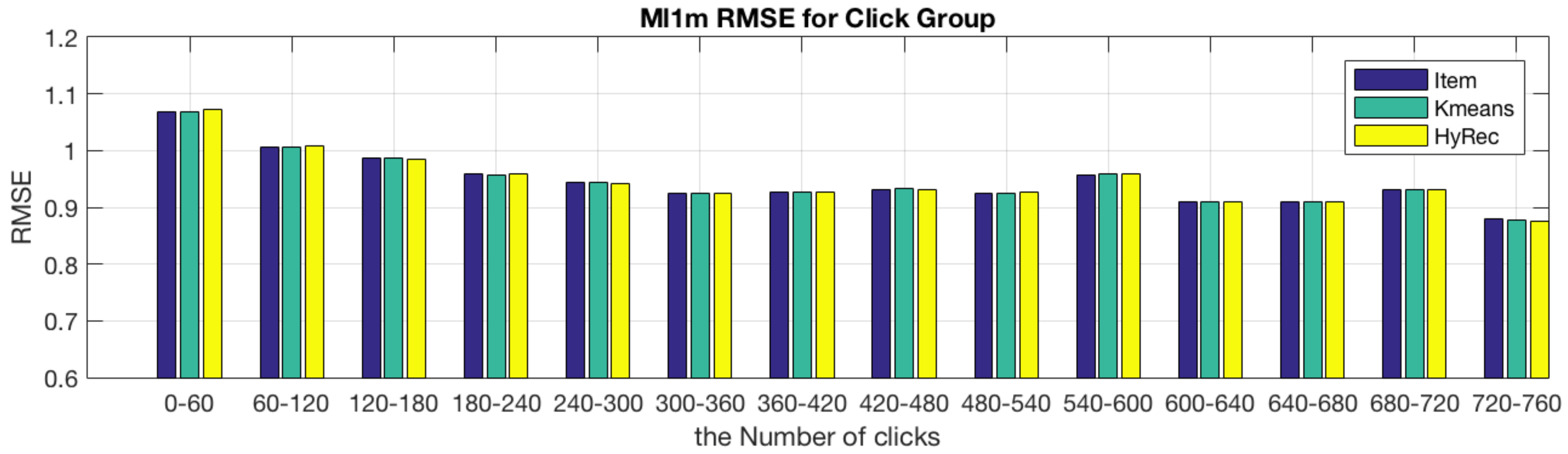
Predication Ciao

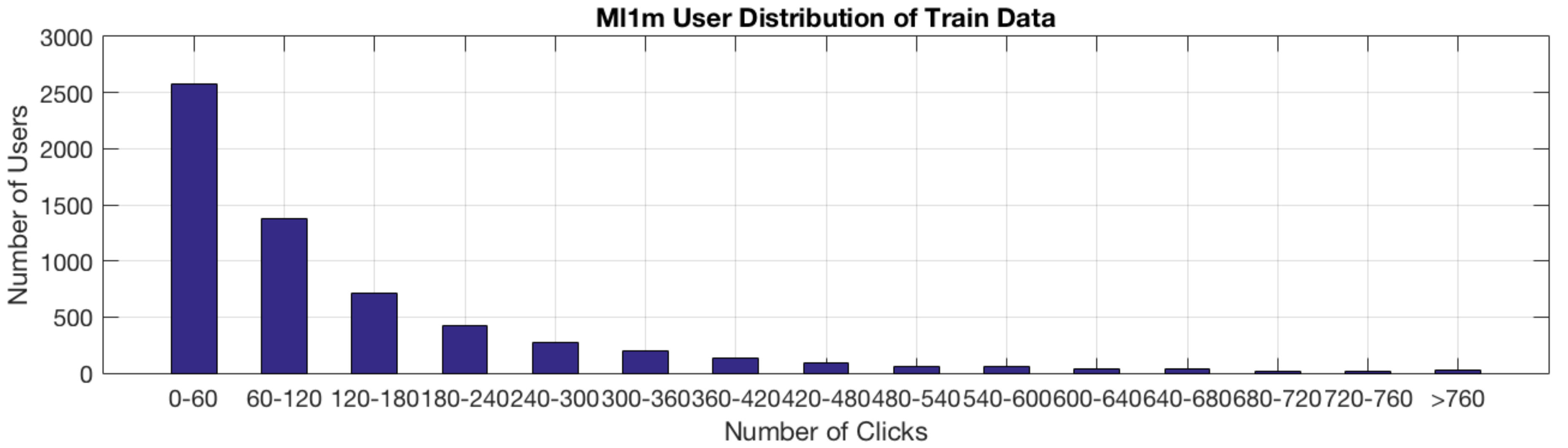


Predication Ml100k

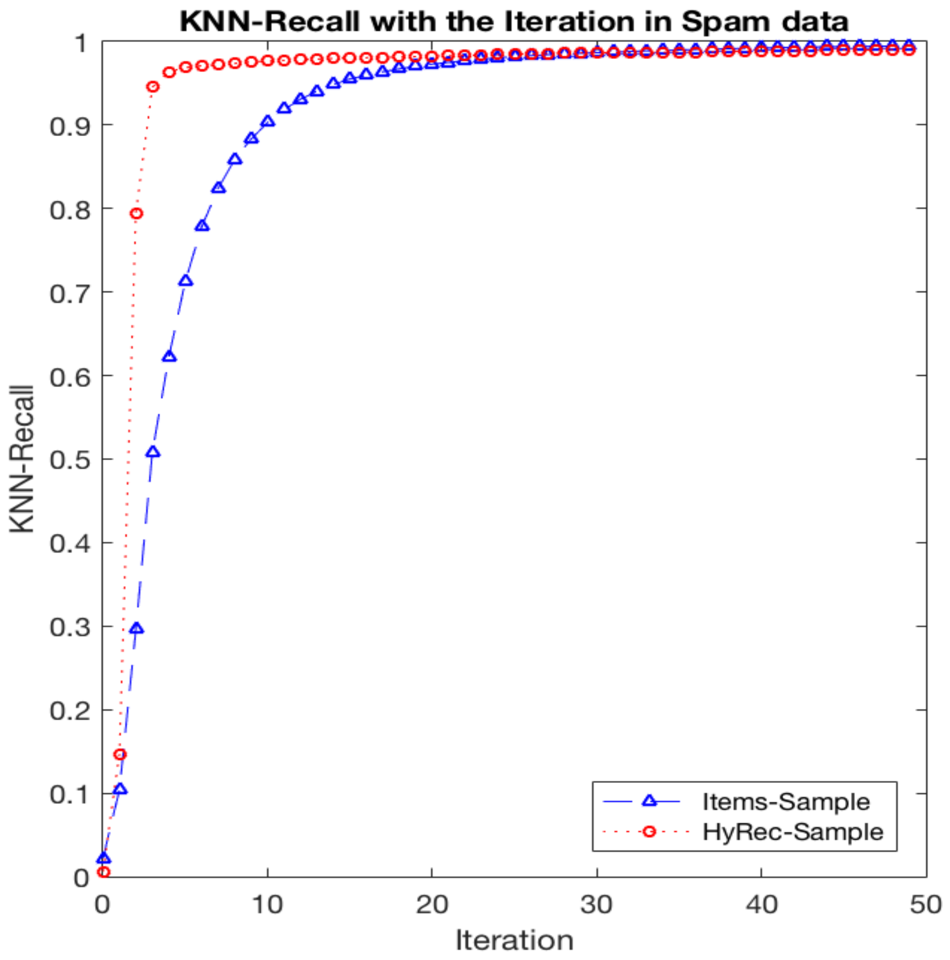


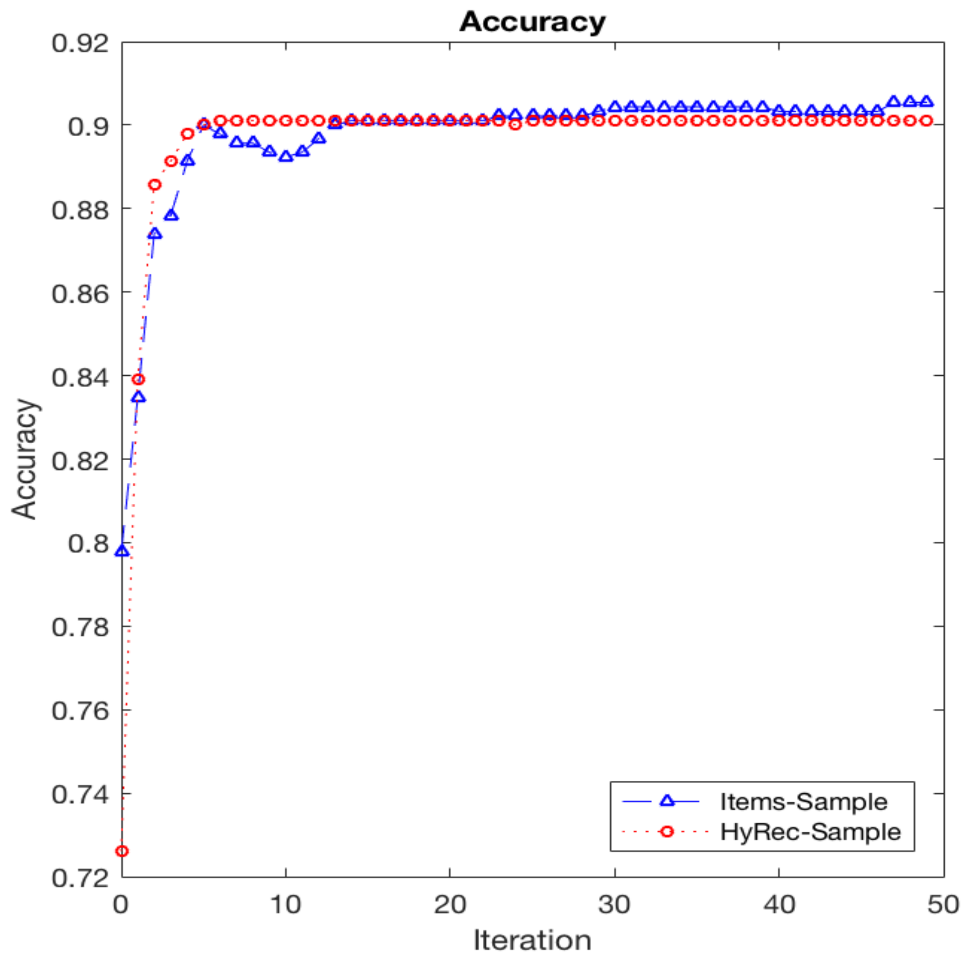
Predication Ml1m





Application On Spam (about 4000 emalis)





Conclusion

- DeKNNs will converge eventually.
- DeKNNs are faster than HyRec.
- It-DeKNN is better than KM-DeKNN
- HyRec can converge faster and is very stable.