Incremental KNN for Scalable Recommendation System [*]

Wei Ma[†]
ma.wei@epfl.ch

Rhicheek Patra[‡]
rhicheek.patra@epfl.ch

ABSTRACT

K nearest neighbor(KNN) is an effective and essential method for the Recommendation System. The complexity of KNN is terrible if the amount of data is huge. The article will demonstrate two online algorithms based on two different cluster methods to approximate KNN graph. They are called K-Means Democratization KNN(KM-DeKNN) and Item-based DeKNN(It-DeKNN). DeKNNs will converge eventually.

1. INTRODUCTION

Recommendation System is pretty useful in practice for the E-Commerce sites. There are two main approaches to produce the recommendation list, Collaborative Filtering and Content based Filtering [4] .Personalization is prevalent among the business companies to provide an accurate and customer-oriented service for the users. Recommendation System should be scalable with the increase of the user data.

With the increases of the users, Recommendation System undertakes growing sharply computing tasks.To alleviate the load of the server, one way is to use more machines or better machines. However, it will give rise to the expensive cost. The other way is to improve the algorithm and the architecture of Recommendation System. The architecture of Recommendation System has three kinds, Centralization, Decentralization and Hybrid.KNN is an efficient algorithm for Recommendation System but its complexity is too high to apply it in the real-time Recommendation System. In the project, we comes up with the two new methods (KMeans-DeKNN and Item-Based DeKNN) to appropriate KNN graph. DeKNNs can be implemented in the distributed environment. It is suitable for the hybrid architecture. Democratization means that any service provider (who has access to multiple resources or who does not have access to any resources) can leverage this kind of approach as the application only uses the clients' machine.

KNN finds the neighbors of one user through computing the similarities with all the other users.In DeKNNs, it does not choose all but choose the potential candidates. Chain Rule is the foundation of DeKNNs. The neighbors of the neighbors are possible to be similar with itself. In the most conditions, the chain rule can work well. However,there are some cases that the chain rule will fail. For instance, there are three users A,B,C. A's profile is (i_1, i_2, i_3), B's profile is $(i_1, i_2, i_3, i_4, i_5)$ and C's profile is (i_4, i_5).A is similar with B and B is similar with C but A is not similar with C.Therefore, in DeKNNs, the other cluster methods are used to find the latent candidates. In my project, I combine the cluster methods with the chain rule to choose the potential candidates.The article comes up with two algorithms ,KMeans-DeKNN(KM-DeKNN) and Item-Based DeKNN(It-DeKNN). KMeans-DeKNN uses online KMeans [3] to classify the users and It-DeKNN uses the items to category the users.

To estimate KNN graph better, the known knowledge should be made full use of. Each user has its own graph that composed by its direct and indirect neighbors. DeKNNs can explore and exploit the graph to find the potential candidates.

The section 2 will show how DeKNNs works. The section 3 is on the experiment and the 4 sections are the conclusion.

2. ALGORITHM

2.1 Choosing Candidates from Neighbors

As the report mentions in the introduction, exploring and exploiting the graph is significant to DeKNNs. When the users have some neighbors, there exits an graph of the relationship among the users in the system. Even though these neighbors are not the top k similar neighbors, it is at least not the negatively similar neighbors. Hence, it has a high probability to choose the latent candidates carefully from

[*]This project focuses on reducing the complexity of KNN and use a common method to predicate the user rate.

[†]Wei Ma comes from China and is a master student in the Computer Science department of EPFL.

[‡]Rhicheek Patra is a PhD student in the Computer Science department of EPFL and works in Distributed Programming Laboratory LPD.He is the supervisor of the project.

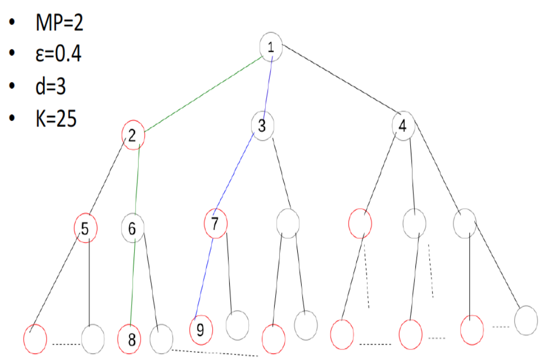

Figure 1: KNN Graph. The circles are removed to make the graph simple and it does not have any influence to express the problem.

the KNN graph to hit the top k neighbors of one user.At first,the circles are removed. If there is a connection forward to the previous node, a new node with the same information will be created as the current node's child. The figure 1 shows the cleaned structure of KNN graph.

When choosing the candidates from the graph, there is one thing that must be prevented. There exist some circles in the graph so the maximum choosing depth should be predefined before searching the candidates. According to the chain rule, the neighbors of the most similar neighbor of one user are possible to be highly similar with the user.However, there are also some scenarios that are converse to the chain rule. Therefore, one parameter ϵ are used to describe when we choose the top one and when we choose another. Its value is between 0 and 1. ϵ is usually set to be less than 0.5 because the chain rule can work well in the most cases. Algorithm 1 shows the details of how to choose candidates from the KNN graph. MP means the mutable paths.It represents the number of searching paths and will vary with the updating results. Algorithm 2 demonstrates how MP changed. n is the number of changed neighbors for one user. $n = |U_{pre} \cap U_{after}|$. U_{pre} is the top K neighbors of the user U before updating and U_{after} is the top K neighbors of the users U after updating.θ is a threshold and its value depends on K. The maximum of MP should be limited to reduce the size of the candidates as well.In the Algorithm 2, MP is limited between 1 and $a \times log_2K$. a is usually defalut 1.

2.1.1 *Example*

Each node in the figure 1 is a users.The children are the top K neighbors of the parent. From the left to the right, the neighbors ranks according to the descending similarity. Now we choose the candidates for the user at node 1. Due to MP is equal to 2, there is two routes to find the candidates. The green path and the blue path are selected in the figure 1. For the green path, it chooses the most similar node 2 with the probability 0.6. In the node 2, it chooses the node 6 with the probability 0.4. Then in the node 6, it choose the most similar node 8. Now the searching depth has arrived at 2 so the searching process stops and returns the neighbors of node 8. The same thing is for the blue path.

```
Algorithm 1: Choose Candidates from the KNN graph
Input: The user's neighbors, noted as($U_i, P$)
Output: Candidate Set,noted as $D$
1  $D \leftarrow empty$
2  $l \leftarrow 0$
3  while $l \le MP$, $a \times log_2K$ is the maximum $MP$ search paths do
4      $l{+}{+}$
5      $d$ =random($c \times log_2K$)
6      for $i \leftarrow 1$ to $d$, $c \times log_2K$ is the maximum $d$ search depth do
7          if $p > \epsilon$ then
8              $U_i$ =the most similar user in $U_{i-1}$'s K nearest neighbor list
9          else
10             $U_i$ = sample one user from $U_{i-1}$'s K nearest neighbor list
11     $D \leftarrow D\cup$ the K nearest neighbors of $U_i$
12 return $D$
```

```
Algorithm 2: MP
Input: $MP$ and the number of the changed neighbors,$n$
Output: new $MP$
1  if $n < \theta$ then
2      $MP$ =min($MP + 1, a \times log_2K$)
3  else
4      $MP$ =max($MP - 1, 1$)
5  return $MP$
```

2.2 KMeans DeKNN

KMeans DeKNN(KM-DeKNN) uses online KMeans to cluster the user.Its assumption is that the users in the same cluster have high possibility to become the top K neighbors for each other. Algorithm 3 shows the process of how KM-DeKNN chooses the candidates. KM-DeKNN has nine main steps.

- Step 1: update the profile of the users related to the record.
- Step 2: update the clusters according to Algorithm 4.
- Step 3: apply Algorithm 1 choose the candidate set.
- Step 4: choose candidate set from the whole users.
- Step 5: choose candidate set from the cluster.
- Step 6: aggregate the candidate sets from step 2,4 and 5 to one set.
- Step 7: send the candidate set in step 6 to the client.
- Step 8: the client will compute the similarity between the target users and the candidates. The client selects the top K candidates and send their id to the server.
- Step 9: the server will update KNN graph according to the returning result.

After we have the KNN graph, the left task is produce the recommendation item list to the users. This content exceeds the project so it will be not talked about in the article. One common method is to recommend the items that the top K neighbors have but the user does not have.

Algorithm 3: Choose Candidates with K-Means

```
Input: The user's profile, noted as($U_i, P$) , the clusters'
       profiles, noted as ($C_i, P$) and the new data,noted as
       $x^t$
Output: Candidate Set,noted as $D$
1  update the user's profile ,noted as $U^t$ according to $x^t$
2  $D \leftarrow empty$
3  update the K-Means according to $U^t$
4  $C \leftarrow$ the current user's K nearest neighbors
5  $U_0 \leftarrow$the current user
6  $l \leftarrow 0$
7  while $l \leq MP$, $a \times log_2K$ is the maximum $MP$ search
     paths do
8      $l++$
9      $d$ =random($c \times log_2K$)
10     for $i \leftarrow 1$ to $d$, $c \times log_2K$ is the maximum $d$ search
         depth do
11         if $p > \epsilon$ then
12             $U_i$ =the most similar user in $U_{i-1}$'s K
                 nearest neighbor list
13         else
14             $U_i$ = sample one user from $U_{i-1}$'s K
                 nearest neighbor list
15     $D \leftarrow D\cup$ the K nearest neighbors of $U_i$
16 $A \leftarrow$ sample K users from the global user table
17 $B \leftarrow$ sample K users from the current user's cluster in
     K-Means
18 $D \leftarrow D \cup A \cup B \cup C$
19 return $D$
```

2.2.1 Online KMeans

KM-DeKNN uses the online KMeans to cluster the users. Algorithm 4 show the process of the online KMeans. When there is a new record, KMeans will update the cluster according to the new profile of the user related to the record. The learning rate will decrease with the increase of time and the number of the users.Its updating complexity in total is the linear time.

When applying KMeans, there are some problems that we must consider carefully. The number of the clusters can be decided according to the types of the users' preference in the filed work. The initialization of the clusters have an important influence on the performance of KMeans. The worst case is that some clusters contains few users because of their bad initial value.The common initial method is taking the average of the sample users set. Another significant thing is whether the data is normalized. It depends on the domain of the features in the dataset.

Algorithm 4: online K-Means

```
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
1  $C_j \leftarrow$find the most similar cluster with the $U^t$
2  learning rate, $r = \frac{1}{|C_j|+1}$
3  update the cluster's center, $C_j = C_j + (U^t - C_j) \times r$
```

2.3 Item-Based DeKNN

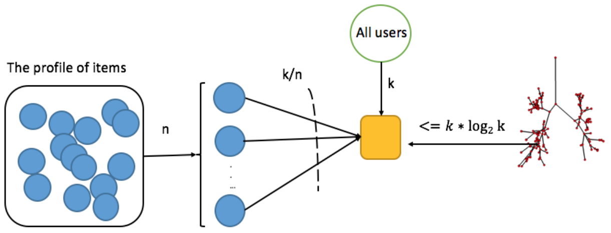

Figure 2: It-DeKNN chooses candidates

KM-DeKNN cannot deal with the cold starting problem. Onilne KMeans has a fixed number of the clusters, which leads to the weakness at finding the new preference among the users. The other thing is that the clusters will become huge and stubborn with the growth of the amount of the users. Item-Based DeKNN(It-DeKNN) can solve some above problems. It is based on the observation that the positively similar users have the common items. It can give a high-quality candidate set for a completely new user. If there is a new item, there is a new cluster.

It-DeKNN has almost the same process with KM-DeKNN. The only difference is that replacing KMeans with Item-based cluster method. Hence, step 2 is changed to update the item cluster according to Algorithm 5 and step 5 is changed to choose the candidates from the item cluster. Algorithm 6 shows the integral It-DeKNN how to choose the candidates. It-DeKNN doesn't use all the item list that the user bought before. It will select a few from them and then take the equal number of the candidates from the selected item lists. Figure 2 shows the choosing candidates process of It-DeKNN.

Algorithm 5: online Item classification

```
Input: A new data at time t,noted as $x^t$ and items'
       clusters's profile, noted as $(C, P)$
Output: update the item classifier
1  $id = theitemidinx^t$
2  put user $id$ in $x^t$ to the profile of $(C_i, P)$
```

2.4 Analysis on Complexity and Convergence

In practice, computing the similarity among the users needs much cost. That is also one reason why the true KNN is not practical to be used in the commerce Recommendation System. DeKNNs will produce the candidate set that has the low bound $3K$ and the up bound $2K + logK$,which is much less than true KNN. The random method is used in DeKNNs to avoid the local optimization. Each user has a probability to be selected as the candidates for the other users so the system will converge after enough time. The neighborhood of u,N_u, converges towards the ideal one (N_u^*).[2]

3. EVALUATIONS

Algorithm 6: Choose Candidates with Items

```
Input: The user's profile, noted as($U_i, P$) , the clusters'
       profiles, noted as ($C_i, P$) and the new data,noted as
       $x^t$
Output: Candidate Set,noted as $D$
1  update the user's profile,noted as $U^t$ according to $x^t$
2  $D \leftarrow empty$
3  $A \leftarrow$ sample K users from the global user table
4  update the item profile according to $x^t$
5  while $i \leq t = a \times log_2(Num_{item})$ the number of the items
     clusters that are used in this time do
6      $i++$
7      $I \leftarrow$ choose $\frac{K}{t}$ samples randomly from $(C_i, P)$
8      $D \leftarrow D \cup I$
9  $B \leftarrow$ the current user's K nearest neighbors
10 $U_0 \leftarrow$the current user
11 $l \leftarrow 0$
12 while $l \leq MP$, $a \times log_2 K$ is the maximum $MP$ search
     paths do
13     $l++$
14     $d$ =random($c \times log_2 K$)
15     for $i \leftarrow 1$ to $d$, $c \times log_2 K$ is the maximum $d$ search
         depth do
16         if $p > \epsilon$ then
17             $U_i$ =the most similar user in $U_{i-1}$'s K
                 nearest neighbor list
18         else
19             $U_i$ = sample one user from $U_{i-1}$'s K
                 nearest neighbor list
20     $D \leftarrow D\cup$ the K nearest neighbors of $U_i$
21 $D \leftarrow D \cup A \cup B$
22 return $D$
```

In this section, we will show that how the new algorithms perform and we also apply the new algorithms to recognize the spam emails. To empirically validate the effectiveness of the proposed approximate kNN, four data sets are used, Ml100k, Ml1M , Ciao and Spam. The primary is to verify : **a)** DeKNNs should converge eventually. **b)** DeKNNs should be faster than HyRec. **c)** DeKNNs have smaller candidate set than HyRec does,which leads to the smaller storage in the client.
In the experiment, K is set to 25. ϵ is set to 0.5,θ is set to 6. The seaching depth d is set to 5. The number of the clusters of KMeans is set to 4. HyRec [2] will be used to compare DeKNNs. The three algorithm were implemented by the two ways, offline and online. The It-DeKNN and HyRec are used to label the spam. At first, the report will describe the features of the dataset that are used. The users are grouped to the different buckets according to their clicks. Each record means a click. The estimation methods are introduced by following.

3.1 Data Description

Table 1 shows the total records and the users in the three dataset Ml100k, Ml1m and Ciao.We also analyze the frequency that the users appears in the three data set. Each record is called a click. The majority of the users in ML100k is grouped to the first 0-60 clicks group as we can see it from the figure 4. Ml1m is more balance than Ml100k.The most of the users have less than 420 clicks.Ciao Dataset is sparser compared with Ml100k and Ml1m. The number of the users in Ml100K has 943 users and it is a little smaller for the

Table 1: DataSet

Name	Users	Records	Rating range
`Ml100K`	943	100,000	1-5
`Ml1M`	6040	1,000,209	1-5
`Ciao`	2210	36065	1-5

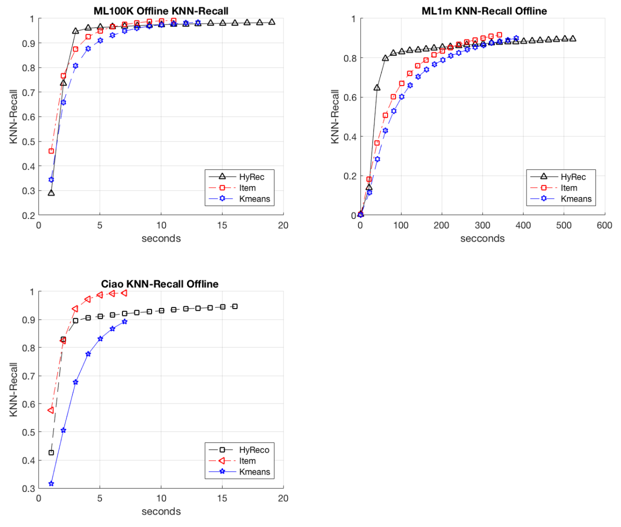

Figure 3: Offline DeKNNs with time

random method because the force KNN is suitable in this case.

3.2 Performance

To estimate how the algorithm approximate KNN graph, KNN-recall is used in the performance.

$$KNN - recall = \sum_{u \in U} \frac{|\widehat{knn_u} \cap knn_u|}{k} / |U| \qquad (1)$$

$\widehat{knn_u}$ is the estimated KNN graph and knn_u is the true KNN graph. $|U|$ is the number of the totla users. To predicate the user rate for a movie, equation 2 is used. The high KNN-Recall means that the algorithm has a good performance and vice versus.

The formula 2 [1] shows how to predicate the user rates for some items.

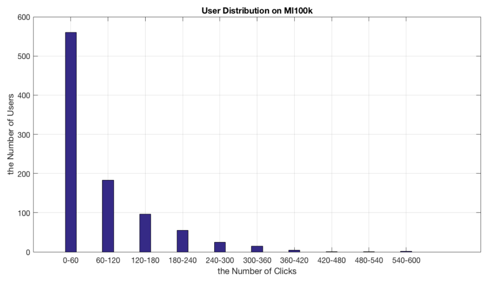

Figure 4: Ml100k User Distribution

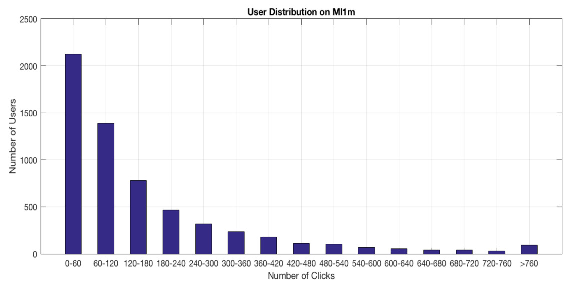

Figure 5: Ml1m User Distribution

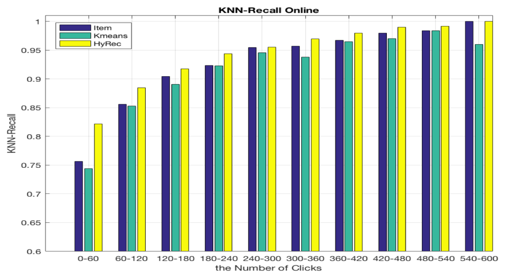

Figure 7: Online Ml100k KNN-Recall

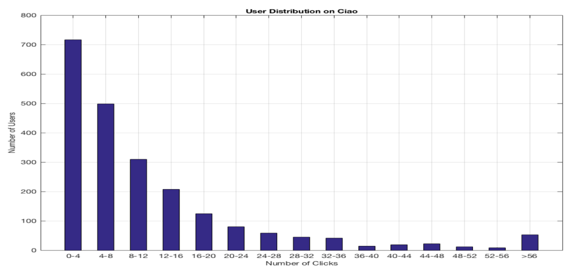

Figure 6: Ciao User Distribution

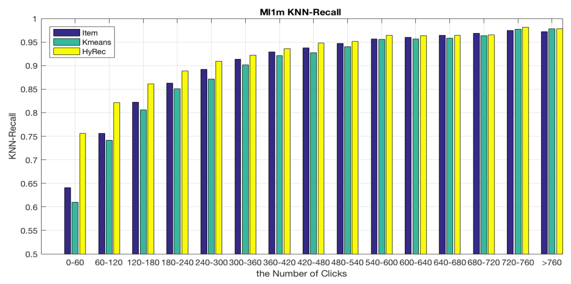

Figure 8: Online Mlm KNN-Recall

$$\widehat{r}(u,i) = \overline{r}(u) + C_0 \sum_{v \in N_k(u,i)} sim(u,v) \times (r(v,i) - \overline{r}(v)) \quad (2)$$

C_0 is an constant and is often set to 1/100 but in our algorithm it is set to 1/200 after some trials.$\widehat{r}(u,i)$ is the predication rate. $\overline{r}(u)$ is the average rating of the user u on the items that he rates.
RMSE(Root mean squared error) is often used to measure how many the error is between the true value and the predication.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\widehat{r}_t - r_t)^2} \quad (3)$$

3.3 Offline DeKNNs

We implemented the three algorithm offline version. Offline means that the three algorithms will know all the data from the beginning. Figure 3 shows the KNN recall of DeKNNs and Hyrec on the three dataset. HyRec has the more stable performance than DeKNNs. It can converge fast. The KM-DeKNN is the worst algorithm compared with others. DeKNNs runs faster than HyRec. Even though it converge slower than HyRec, it will exceed HyRec in the end.

3.4 Online DeKNNs

Online means that at the beginning, the algorithms do not know anything about the data. The data will comes and be as the input the algorithm with time. We compute the recall of each group according to the user distribution. We find that the users that have a high number of clicks have high recall. Figure 7,8 and 9 turn out the point.The dataset is splited to two part, 80% for the training set and 20% for the test set. Figure 10,11 and 12 show the predication RMSE error for each group on the user distribution of the training data set.Overall, RMSE will decrease with the increase of the clicks.

3.5 Storage Requirement

We also analyze the storage of each algorithm.The candidate size is use to measure the storage.Figure 13 shows the result. The candidate size will increase with the growth of K. HyRec rises obviously and needs much larger candidates than DeKNNs. Figure 14 shows how many bytes the algorithms require in the client machines. It illustrates that HyRec need much more storage than HyRec.

3.6 Application on Spam

We use offline It-DeKNN sample method and Hyrec sample mehtod to classify the emails. The spam data set has 4601 emails. The data set is split to two part, 80% for the training data and 20% for the test data. The first step is to find the top k neighbors of each email. Then next is voting to decide whether the email is spam or not.
We use four methods to predicate. Then the four methods votes together.The figure 15 shows the voting process. The

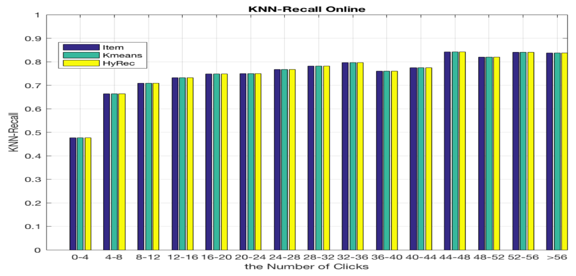

Figure 9: Online Ciao KNN-Recall

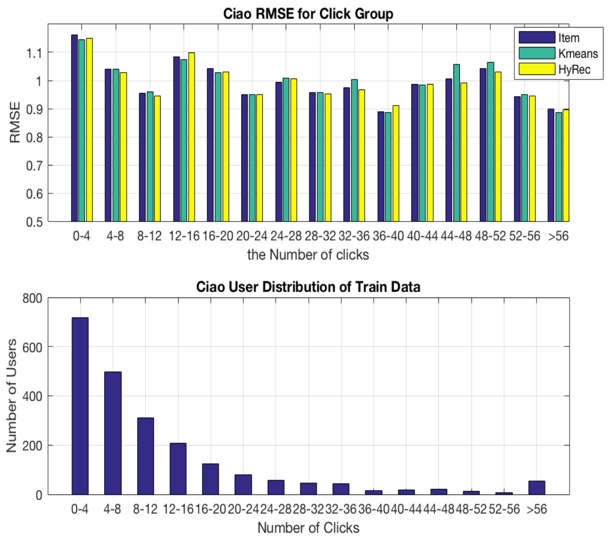

Figure 10: Online Ciao Predication

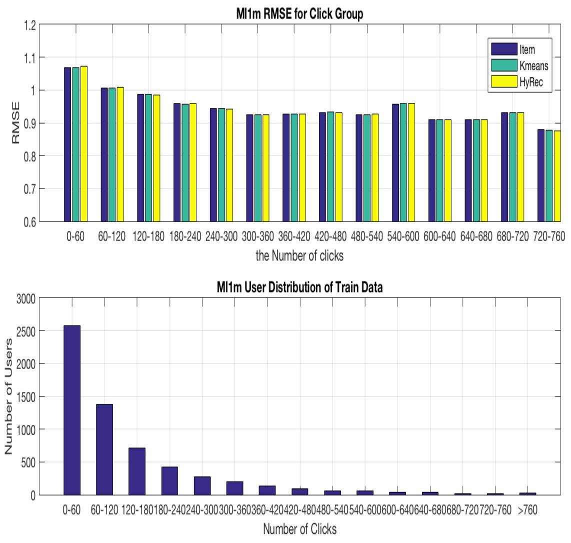

Figure 12: Online Ml1M Predication

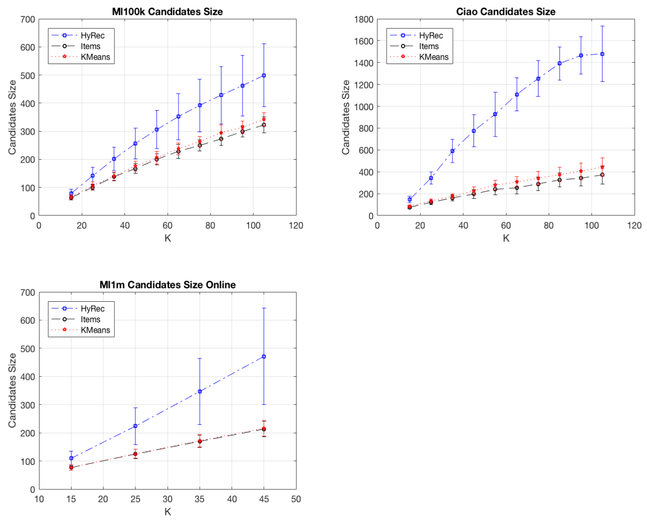

Figure 13: Candidate Size

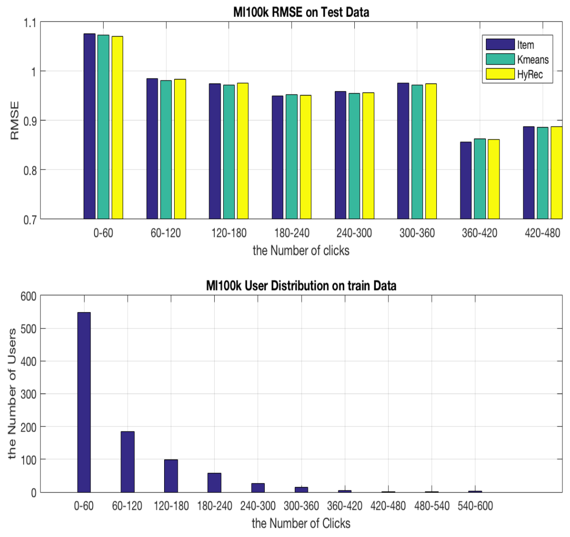

Figure 11: Online Ml100k Predication

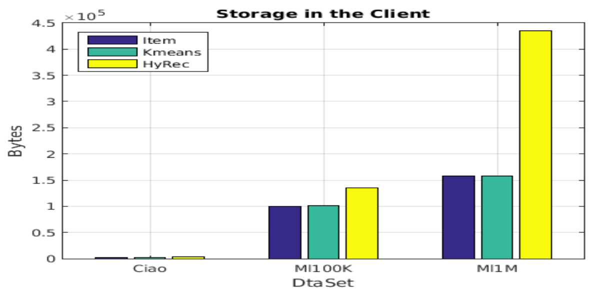

Figure 14: Storage in the Client

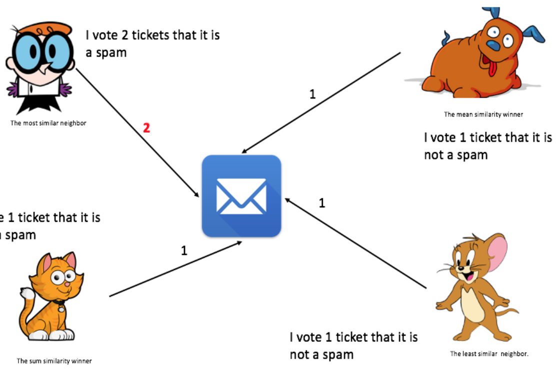
I vote 2 tickets that it is a spam
The most similar neighbor
2
1
The mean similarity winner
I vote 1 ticket that it is not a spam
I vote 1 ticket that it is not a spam
1
1
The sum similarity winner
I vote 1 ticket that it is not a spam
The least similar neighbor.

Figure 15: Voting to decide whether it is a spam or not

P\T	1	0
1	tp	fn
0	fp	tn

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

Figure 16: Accuracy Equation

spam gets 2 tickets and the non spam gets tickets. Hence, the email is not a spam.

- 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 have one ticket.

It is a binary classification problem. The recall is used here and we will compute the predication accuracy. Figure 16 shows how to compute the predication recall. tp is the correct result for $1.fp$ is the wrong resutl for 0. tn is the correct result for 0. fn is the wrong result for 0.
Figure 17 shows the KNN-Recall and the accuracy result with the iteration. Overall, the accuracy increases with the growth of KNN-Recall.

4. CONCLUSION

DeKNNs is faster than HyRec but HyRec can converge faster. DeKNNs need much less storage in the client than HyRec. DeKNNs will increase the load the server because it need to storage the cluster information. HyRec is very stable. KM-DeKNN has the worst performance among the three method. There are tradeoff between the accuracy and the time and space. To get a better approximation of KNN graph, DeKNNs and HyRec store the intermediate result to reduce the time. DeKNNs use the cluster method to reduce the candidate size and at the time get a better candidate size. It needs more storage to record the cluster information in the server than HyRec. HyRec needs more candidates and converges faster than DeKNNS. DeKNNs run faster but it need a longer time to converge.

In the future work, more clusters methods should be tried and we also need consider the privacy of the users because the computing task is completed in the client machines.

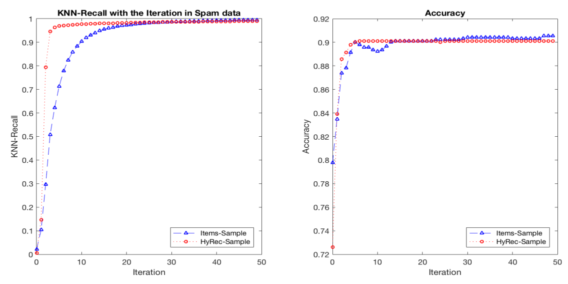

Figure 17: KNNRecall and Accuracy

5. REFERENCES

[1] A. Bellogín, P. Castells, and I. Cantador. Neighbor selection and weighting in user-based collaborative filtering: a performance prediction approach. *ACM Transactions on the Web (TWEB)*, 8(2):12, 2014.

[2] A. Boutet, D. Frey, R. Guerraoui, A.-M. Kermarrec, and R. Patra. Hyrec: Leveraging browsers for scalable recommenders. In *Proceedings of the 15th International Middleware Conference*, pages 85–96. ACM, 2014.

[3] A. King. Online k-means clustering of nonstationary data. *Prediction Project Report*, 2012.

[4] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.