Efficient personalized recommendation of mobile web content using an EM-based clustering method

Ming He¹,², Alvin Chin², Enhong Chen¹, Jilei Tian²

¹School of Computer Science, University of Science and Technology of China
²Xpress Internet Services, Microsoft, Beijing, China

¹,²yuminghe@vip.163.com, ²{alvin.chin, jilei.tian}@microsoft.com, cheneh@ustc.edu.cn

Abstract—Many applications recommend personalized content to users based on their interests. However, personalized recommendation is time and memory consuming especially for commercial systems that have huge numbers of users, requests and big data that require complex computation. Since users do not totally have unique interests, we can cluster similar users then recommend the same items to users belonging in the same cluster. Even though clustering-based recommendations are efficient, the recommendation items to users may not be accurate. We present an Expectation-Maximization (EM)-based personalized recommendation method for selecting the appropriate items efficiently and accurately. We use a browser log dataset to compare our method with personalized, k-means, and EM-based recommendations according to average rank, novelty, diversity, and time performance. Results show that based on average rank, novelty and diversity, our proposed method performs close to that of personalized, however it is less efficient than k-means. Since k-means has the worst average rank, novelty and diversity, our method is the best overall.

Keywords—Personalized recommendation; EM clustering; k-means clustering; mobile content recommendation

I. INTRODUCTION

Many applications and services recommend content to users based on their interests. For example, Amazon recommends books based on your purchasing and viewing history that is similar to others. Flipboard recommends interesting news based on your interests and your friends. The recommendation algorithms used are based on collaborative filtering or content-based filtering.

Personalized recommendation recommends items to users that match a user’s interests, however it is time consuming and inefficient [15]. Since users do not totally have unique interests, we can cluster similar users based on their behavior, then recommend the same items to users that belong in the same cluster. Clustering algorithms such as k-means and EM can be used for recommendations, which decrease the computation time greatly, but loses the recommendation accuracy [13]. How to select the number of clusters, how to select the number of items from each cluster, and how to combine the items to form the recommendations are still unresolved issues [20].

In this paper, we propose an efficient EM-based personalized content recommendation model. We obtain the topic distribution for each content item and each user using LDA, then use EM clustering to obtain the user’s cluster probability distribution. We calculate the number of optimum clusters for each user and the number of content items for each cluster to ensure that we have enough and unique items for recommendation. We select the content items from ranking the similarity scores between the item’s topics and the user’s topics, and recommend the highest items to the user.

Using the browsing log of users from Nokia Xpress Browser [17], we compare our method with personalized, k-means and EM-based recommendations according to our evaluation criteria of average rank, novelty, diversity and recommendation time. Results show that our method’s average rank is close to that of personalized, diversity is close to that of personalized, novelty is better than k-means and personalized, and recommendation time is a little longer than k-means. Since k-means has the worst average rank, novelty and diversity, therefore our method is considered the best overall compared to the existing cluster-based recommendations. Our model can be applied to every content-based personalized recommendation, such as news recommendation, book recommendation, application recommendation and so on.

Our contributions are the following. First, we propose an improved recommendation method over personalized and clustering-based methods. Second, we develop an algorithm for selecting the clusters and content items that efficiently minimizes the time. Third, we propose four metrics for evaluating the recommendation methods.

The outline of this paper is as follows. Section 2 presents an overview of related work and we describe our dataset in Section 3. In Section 4, we describe the recommendation models used for our evaluation. Our results are described in Section 5. Finally, in Section 6, we conclude the paper and discuss areas for future work.

II. RELATED WORK

Recommendation algorithms can be categorized into collaborative filtering and content-based systems.

Collaborative filtering (CF) can be divided into memory-based CF and model-based CF. Memory-based CF algorithms are easy to implement and have good performance. Memory-based CF on dimensionality reduced rating data can generate more accurate predictions than on the original sparse rating data [8]. Disadvantages of memory-based CF are their...
dependence on user ratings, data sparsity, cold start and scalability [12]. Many model-based CF techniques suffer the problem of the model-building expense being high [21].

Content-based recommender systems learn a profile of the user’s interests based on the features present in items the user has judged [18], however, they are limited by those features [4]. To address the problem, Basu et al [2] add features to examples that indicate the identifiers of other users who like an item and Ripper et al [7] use this to learn profiles with both collaborative and content-based features.

Besides CF and content-based recommendations, we can consider recommendations based on clustering. Recommendation algorithms based on k-means clustering can improve time efficiency [10], address data sparsity [16], and find relevant items [6]. Other clustering methods such as hierarchical clustering [22] can improve accuracy and timeliness. Even though EM clustering [9] can be used to further improve recommendation accuracy, efficiency suffers compared to k-means. Although clustering-based recommendations can improve recommendation efficiency, it does not consider a user’s personal habits.

Our paper proposes an efficient clustering-based recommendation method using EM clustering to improve both the recommendation efficiency and the accuracy of a user's personalized recommendations.

III. RECOMMENDATION MODEL

In this paper, we create 4 recommendation models, where the first 3 models are based on previous work and the last model is our proposed new model. We use the term item to refer to content, product, service or any type of object that can be recommended and consumed by the user; and the term topic to refer to the subject that uniquely describes or characterizes the item or user. The topic distribution is a collection of topics where each topic has a probability of its relevance to a user or an item, and is obtained from topic extraction methods such as LDA. We recommend items to users based on the similarity of the item with the user’s profile or with another user. Before we describe each of the recommendation models, we introduce the modified cosine similarity which we use to calculate the similarity.

A. Modified cosine similarity

In this paper, we use a modified version of cosine similarity as the similarity measure as shown in Eqn. (1) called the modified cosine similarity.

\[
CS_{ij} = \frac{\sum_{k=1}^{T} t_{ik} * t_{jk}}{|\|t_i\|_1 \cdot |\|t_j\|_1|} \quad (1)
\]

where \(CS_{ij}\) is the modified cosine similarity between user \(i\) and item \(j\), \(t_{ik}\) is the probability of topic \(k\) for user \(i\), \(t_{jk}\) is the vector of topic probabilities for item \(j\), \(T\) is the number of topics, and \(|\|t_i\|_1|\) is the 1-norm of vector \(t_i\). From ranking the similarity scores, we recommend top \(R\) items with high similarity scores to the user.

Modified cosine similarity is a modified version of cosine similarity used in matching algorithms where instead of the traditional 2-norm vector (or Euclidean distance) in the denominator, we use the 1-norm (or linear distance). We select modified cosine similarity because we discovered that cosine similarity was not linear and did not yield accurate results which we explain below.

We discovered from our experiments that using cosine similarity for the cluster-based recommendation methods, gave average rank scores of over 700 which were very poor. Splitting the user’s profile into several clusters with different probabilities, and then combining those clusters to rebuild the user’s profile, should ideally rebuild the user’s profile without any loss. However, we discovered that we lost some information of the user when we clustered users. When we split the user’s profile, it was a linear change. If we used 2-norm distance in Eqn. (1), the computed score was not linear. We illustrate this with an example below.

For example, with 3 clusters and 1 user, we can split the user profile \(u\) into 3 clusters with cluster profile \(c_i\), probability distribution \(p\) and URL profile \(l\) as shown in Eqn. (2).

\[
u = p_1 \cdot c_1 + p_2 \cdot c_2 + p_3 \cdot c_3 \quad (2)
\]

Now, we compute the score between user and URL based on user profile using 2-norm distance in Eqn. (3).

\[
S_1 = \frac{u \cdot l}{|\|u\|_2 \cdot |\|l\|_2|} \quad (3)
\]

The score based on clustering is Eqn. (4).

\[
S_2 = \sum_{i=1}^{3} p_i \cdot \frac{c_i \cdot l}{|\|c_i\|_2 \cdot |\|l\|_2|} \quad (4)
\]

Now, we want to prove whether \(s_1\) is equal to \(s_2\) which obviously from the above is not. However, if we use 1-norm in Eqn. (3) and Eqn. (4), we can prove that \(s_1\) is equal to \(s_2\).

When we get the user’s profile and cluster’s center, we have normalized their profile. The sum of its topic probability distribution is 1. If we use 1-norm in Eqn. (3) and Eqn. (4), giving \(s_{11}\) and \(s_{21}\) respectively, \(|\|u\|_1|\), \(|\|l\|_1|\), \(|\|c_i\|_1|\), \(|\|c_2\|_1|\) and \(|\|c_3\|_1|\) are all 1. Then, we obtain

\[
s_{11} = \frac{u \cdot l}{|\|u\|_1 \cdot |\|l\|_1|} = u \cdot l
\]

\[
s_{21} = \sum_{i=1}^{3} p_i \cdot \frac{c_i \cdot l}{|\|c_i\|_1 \cdot |\|l\|_1|} = \sum_{i=1}^{3} p_i \cdot (c_i \cdot l) \quad (5)
\]

Combined with Eqn. (2), \(s_{11}\) equals \(s_{21}\) in Eqn. (5). After clustering, if we used 1-norm in Eqn. (1), we found that the recommendation results using average rank were more accurate than the recommendation results based on 2-norm.

We now explain each of the recommendation models below.
B. Personalized Recommendation

Personalized recommendation recommends relevant items to a user based on whether those items match the user’s profile where the user’s profile is described as a user’s topic distribution. To determine relevant items for a user, we calculate the similarity between the item’s topic distribution and the user’s topic distribution using the modified cosine similarity measure in Eqn. (1).

C. k-means-based Recommendation

k-means-based recommendation uses k-means to first cluster the users based on a user’s topic distribution and then uses the item’s topic distribution of the clustered users to create a cluster profile of items. We calculate the modified cosine similarity between each item’s topic feature and each cluster center using Eqn. (6), rank items for each cluster by the similarity scores between items and the cluster, and select the top R items to the users who belong to this cluster.

Eqn. (6) shows the modified cosine similarity for this,

\[ kS_{ij} = \frac{\sum_{k=1}^{K} c_{ik} \cdot t_{jk}}{||c_i||_1 \cdot ||t_j||_1} \]  

(6)

where \( kS_{ij} \) is the modified cosine similarity for k-means between cluster \( i \) and item \( j \), \( c_{ik} \) is the probability of topic \( k \) for cluster \( i \), \( c_i \) is the vector of topic probabilities for cluster \( i \), \( t_{jk} \) is the probability of topic \( k \) for item \( j \), \( t_j \) is the vector of topic probabilities for item \( j \), \( T \) is the number of topics, \( ||c_i||_1 \) is the 1-norm of vector \( c_i \), and \( ||t_j||_1 \) is the 1-norm of vector \( t_j \).

D. EM-based Recommendation

EM-based recommendation clusters the users by the EM approach over the latent topic space [9], and calculates the cluster center by averaging multiplications between the user’s topic feature and user’s probability in the cluster. We calculate the modified cosine similarity score as in Eqn. (6) between each item’s topic feature and each cluster center, and rank the similarity scores for each cluster. We calculate the new item’s score by multiplying the user’s cluster probability and item’s similarity score for all clusters, and reorder items by the new score to recommend R items to the user.

E. Efficient EM-based Personalized Recommendation

If we have many clusters, and we select all clusters and all items for EM-based recommendation, even though the recommendation results would be the most accurate, it still could be inefficient for recommendation because of the long computation time. In commercial recommendation systems, it is imperative that items are recommended efficiently to users, since users do not have patience to wait for the results. We can choose selected clusters and selected items for merging, however how many and what to choose? From the selection, there could be duplicate items from the selected items from selected clusters.

To solve these problems, we propose efficient EM-based personalized recommendation to leverage the probability that a given user is assigned to each candidate cluster and its estimated duplication rate among those data. Our method is similar to EM-based recommendation except for the following steps which are explained below.

1) Select top m clusters for user

We select top \( m \) clusters for a user \( u \) using the following criteria:

(i) EM cluster probability calculated between user and each cluster \( c \) (designated as \( p_c \)) is larger than the threshold \( T_1 \) and

(ii) cumulative EM cluster probability between the user and selected clusters does not exceed threshold \( T_2 \).

The worst case for selecting clusters would be selecting all the clusters which is exactly the same as baseline EM, and would occur when each cluster’s probability \( p_c \) is the same which is \( 1/|C| \) where \( C \) is the total number of clusters. Since we want to improve over EM, therefore we select those clusters whose cluster probability is greater than \( 1/|C| \). As a result, we choose \( 1/|C| \) as the threshold \( T_1 \), and \( T_2 \) helps to select which cluster to include in the top \( m \) clusters.

However, if \( |C| \) is large, then using \( T_1 \) may still result in too many clusters. In the premise of ensuring the accuracy of the model, we want to reduce the number of clusters obtained from \( T_2 \) in order to improve the efficiency. Therefore, we introduce the cumulative EM cluster probability threshold \( T_2 \) to reduce the number of clusters obtained from \( T_1 \). We continue selecting clusters until the cumulative EM cluster probability approaches the threshold \( T_2 \). The reason why we choose the cumulative EM cluster probability is because cumulative probability (in our opinion) is the easiest way to ensure that enough relevant clusters are chosen and that the recommendation efficiency is high.

We do not want to have too few clusters otherwise recommendation efficiency and accuracy will suffer. In addition, we do not want to select all clusters because recommendation efficiency will suffer, so we need to select something in between. This is guaranteed by \( T_2 \) which is the cumulative cluster probability. We want to make sure that we have enough clusters covered to ensure that we have good recommendation efficiency. After that, we can ensure the accuracy of the model and improve the performance.

In summary, \( T_1 \) ensures that good clusters with high probability are chosen and that poor clusters with very low probability are not, while \( T_2 \) determines the number of clusters \( m \) to select. The cluster selection is governed by the ordering of clusters in descending order of EM probability, to guarantee that the best clusters with the highest probability are included. We explain how we calculated \( T_1 \) and \( T_2 \) in our experiments, which are described in Section IV.

The complete details on how we select the top \( m \) clusters for a user based on the above, is explained in Algorithm 1.

2) Select top R items from selected clusters for recommendation

If we compute the similarity scores of all items in a selected cluster, it takes too much time. So, we can reduce the number of recommended items by ranking their similarity scores from selected clusters. The number of top recommended items \( R \) is
governed by the number of recommendations to display to the user in the application. However, how many more items should we look for until we get exactly \( R \)? If we choose exactly \( R \) items, there is the possibility that we may get less than \( R \) items, because selected items from different clusters may duplicate, thus reducing \( R \) and affecting the quality of the results. Therefore, we cannot directly select the top \( R \) items from the clusters. To solve this problem, we compute the duplication rate \( (dr) \) which helps us to determine the additional number of items to select to compensate for the duplication items in order to get to \( R \).

\[
dr = \frac{r}{(|C|-1)+r} \quad (7)
\]

The duplication rate can then be plotted to find a fitting equation. Then, the new adjusted \( R \) is \( R_{adj} = R + r \). We can then select the top items from each cluster (by similarity score) based on the user’s probability in the cluster, and combine them together to form the recommended list of items.

However, this approach may not be the most optimal because we may not get a combined \( R \) items from all clusters due to duplication, thus requiring us to select other lower ranked items in each cluster. This could decrease the recommendation efficiency. Thus, we add a margin coefficient \( mc \) to \( r \) to decrease the iteration times, and improve recommendation efficiency. Therefore, the new equation of \( R_{adj} \) is Eqn. (8) below.

\[
R_{adj} = R + (1 + mc) \cdot r \quad (8)
\]

The selection of \( mc \) can be calculated by performing experiments on the dataset to obtain the minimum items efficiency. The entire procedure is detailed in Algorithm 2.

### IV. EVALUATION AND EXPERIMENTS

We implemented the EM clustering and our proposed recommendation method using the Scikit-learn tool [19]. The other methods and the evaluation we implemented the code ourselves.
A. Dataset and Data Preprocessing

1) Dataset description

For testing the different recommendation methods in Section III, we used approximately 2 months of selected data from the web browsing log of users from Nokia Xpress Browser [17], which is a mobile web browser for Nokia mobile phones, as our dataset. The user base is over 100 million users. The browsing log contains URLs that users visit from the browser, along with the anonymized device ID and time of access. We did not identify the user’s identity and safeguarded user’s privacy in accordance with company and legal policies. Detailed statistics of the selected data that we collected are shown in Table I.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting Time</td>
<td>2/24/2013</td>
</tr>
<tr>
<td>Ending Time</td>
<td>4/15/2013</td>
</tr>
<tr>
<td>Total # of URLs Before Filtering</td>
<td>1161757</td>
</tr>
<tr>
<td>Total # of URLs After Filtering</td>
<td>75956</td>
</tr>
<tr>
<td>Total Users Before Filtering</td>
<td>612066</td>
</tr>
<tr>
<td>Total Users After Filtering</td>
<td>5771</td>
</tr>
<tr>
<td>Average # of URLs Per User</td>
<td>13.14</td>
</tr>
</tbody>
</table>

Since our objective is to recommend news content, we filtered the browsing log to remove any personal URLs and URLs that were not news related, and discovered that the length of some of the users’ browsing history was significantly less than the minimum of 8, thus we removed these users from our dataset as they were not considered to be active users and did not contribute to the recommendation. We used the total number of URLs after filtering |L| and total number of users after filtering |U| (as in Table 1) for conducting the experiments in subsection C.

2) Extracting URL content and topics

Many methods exist for extracting URL content such as BoilerPipe [14] based on DOM tree, and Arc90 [1] based on regular expressions [11]. Methods that use a DOM tree are intuitive and effective, but take polynomial time and suffer the problem of morbid HTML. CX-EXTRACTOR [5] examines the webpage’s block structure to compute the page’s line-block word count distribution function, without needing HTML tags. Since it takes linear time and it is easy to find the body of the webpage, we used CX-EXTRACTOR to extract the URL’s content.

We obtained the topics in the webpage’s content using Yahoo’s implementation of LDA [3], chose 100 topics, and then obtained the URL topic distribution which is a list of topics along with their probabilities. We computed the user’s topic distribution based on the topic distribution over the URLs in the user’s browsing history, and this formed the user’s profile which we used to recommend the URLs to the user according to the methods in Section III.

We now explain the metrics which we used to evaluate our methods. These metrics are designed to evaluate the quality of the content and the relevance of the recommendation to the user.

B. Evaluation Metrics

1) Average rank

Average rank AR is the average position of common URLs CR_u between user u’s recommendations and the user’s browsing URL history. It is calculated in Eqn. (9),

\[ AR = \frac{\sum_{u \in U} \sum_{r \in CR_u} Pos_{ur}}{|U|} \]  

where Pos_{ur} is the position of URL r in the list of user u’s recommendations, CR_u is the set of common URLs between user u’s recommendations and user’s browsing history, U is the set of all users and |CR_u| is not zero. A smaller average rank is considered better than a larger one, because it shows that the content item that the user viewed is recommended higher in the recommended list of content items.

2) Novelty

Novelty N is the average “hotness” of the recommended item. If “hotness” of a URL is higher than another that means that more users viewed this URL than another, and is calculated in Eqn. (10),

\[ N = \frac{\sum_{u \in U} \sum_{r \in R_u} n_{ur}}{|U|*|R_u|} \]  

where n_{ur} is the number of users who select URL r, R_u is the list of recommended URLs for user u and U is the set of all users. The smaller the novelty, the better the evaluation because this means that the URL is not so popular, meaning it is unique.

3) Diversity

Diversity D aims to make different recommendations to different users and is calculated in Eqn. (11),

\[ D = \frac{\sum_{u \in U} \sum_{r \in R_u} |R_u \cap R_{u'}}{|U|*|U'|} \]  

where R_u is the recommended list of URLs for user u, and U is the set of all users. We recommend the same number of URLs to every user, so |R_u| = |R_{u'}| and diversity is in the range of 0-1. A larger diversity means that the recommendation list has more different recommendations for the user which is desirable.

C. Comparison of Recommendation Methods

We evaluated the recommendation methods of Section III according to our evaluation criteria above and used the user’s browsing history as the baseline. For testing the recommendation methods, we used the filtered dataset |L| from Table 1. We used 80% of the user’s log in chronological order (based on this user’s first viewed time to last viewed time within the dataset time duration) for training and 20% of the remaining log for testing. We computed the average rank, novelty and diversity for each recommendation method to evaluate the content quality. Then we computed the
recommendation time for the different recommendation methods to compare the performance.

In a commercial recommendation system, recommended items need to be computed in real time and therefore need to be efficient. We desired the cluster count to be smaller rather than larger because if the cluster count is smaller, then the model will be simpler and faster to compute. Therefore from our dataset, we selected the optimum number of clusters $k$ for k-means and EM taking into account time performance, model generalization and the average rank. As a result, we selected $k = 20$ for k-means and EM.

For EM-based recommendation, we can have different types based on the number of clusters and the number of URLs. For number of clusters, we can choose either fixed clusters or dynamic clusters (each user has different number of clusters) and for number of URLs, we can choose either fixed number of URLs or dynamic number of URLs (each cluster has different number of URLs). Thus, this resulted in four EM methods which we implemented:

1) **EM (all)** – fixed clusters (all clusters $=20$), fixed URLs (all URLs)
2) **EM (top 2)** – fixed clusters (top 2), fixed URLs (all)
3) **EM (mc=20%, top 2 clusters)** – fixed clusters (top 2), dynamic URLs
4) **Efficient EM** – dynamic clusters, dynamic URLs (mc = 20%)

For the fixed clusters, we chose the top 2 clusters otherwise if we had chosen the top 1 cluster, then we would have had similar results to k-means. For efficient EM, we selected the cluster probability threshold $T_2 = 1/20 = 0.05$ because this is where all clusters can have equal cluster probability. We selected the cumulative cluster probability threshold $T_2$ as follows. If $T_2$ is too high, recommendation efficiency suffers (because we have URLs from many clusters to select from), however if $T_2$ is too low, recommendation accuracy suffers (because we have too few clusters to select URLs from).

For efficient EM, by plotting the cumulative cluster probability $T_2$ vs. the average number of selected clusters for each user (denoted by $a$) as shown in Fig. 1, we selected the value of $T_2$ where the average number of clusters per user was greater than 1 (since we wanted to be better than k-means). We discovered when $T_2=0.95$ we obtained $a=1.1$ which meant that there was a 95% probability that users belonged to dynamic selected clusters and 5% probability that users belonged to unsellected clusters. For our experiments, we chose the number of recommended URLs $R$ to be 100.

After calculating the duplication rate in Eqn. (7) for efficient EM, we then obtained the fitted duplication rate which was $dr = 0.0011 + R = 0.0406$ with a 0.996 goodness of fit. When the margin coefficient $mc = 0.2$, we discovered that the performance of efficient EM recommendation was the best, therefore, we selected $mc = 20\%$ for the dynamic clusters.

We now describe the results of our evaluation of the recommendation methods based on content quality and performance.

![Average number of selected clusters with $T_2$](image)

1) **Content quality (average rank, novelty, and diversity)**

Table II shows the results for the 6 recommendation methods (personalized, k-means and the 4 EM-based recommendations) for content quality and shows that personalized had the best average rank and diversity, while EM (top 2) had the best novelty.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average rank</th>
<th>Novelty</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personalized</td>
<td>487.77</td>
<td>4.74936</td>
<td>0.95931</td>
</tr>
<tr>
<td>$k$-means ($k=20$)</td>
<td>683.50</td>
<td>5.37123</td>
<td>0.82134</td>
</tr>
<tr>
<td>EM (all)</td>
<td>518.57</td>
<td>4.72565</td>
<td>0.92903</td>
</tr>
<tr>
<td>EM (top 2)</td>
<td>519.79</td>
<td><strong>4.72465</strong></td>
<td>0.92904</td>
</tr>
<tr>
<td>EM (mc=20%, top 2)</td>
<td>520.33</td>
<td>4.72466</td>
<td>0.92902</td>
</tr>
<tr>
<td>Efficient EM</td>
<td>520.77</td>
<td>4.72556</td>
<td>0.92904</td>
</tr>
</tbody>
</table>

Personalized recommends URLs with high score to each user, so the URLs which the user had viewed have higher probability to be top ranked in the recommendations, thus personalized had the best average rank. What’s more, personalized recommends URLs which are unique, so the diversity of personalized recommendation is better than other methods. Since k-means recommends items to each user ignoring personalization, uses non-overlapping clusters and recommends the same items to users in the same cluster, therefore its novelty, diversity and average rank was the worst. According to the novelty, efficient EM was close to that of personalized recommendation. Based on average rank, novelty and diversity, personalized was the best, followed by EM (top 2), EM (all), EM (mc=20%, top 2), efficient EM, and k-means. The various EM-based recommendation methods had similar novelty, diversity and average rank. However, personalized was not efficient, therefore we computed the performance of each of the methods below.

2) **Performance**

For measuring the performance of the recommendation methods, we calculated the total recommendation time as the total time to recommend items to all users, excluding clustering time and time to compute the user and URL profiles. The average recommendation time was then computed as the total recommendation time divided by the number of all users, and is shown in Table III for all methods. The environment for
performance testing was a PC running Windows 7 operating system with 2 GB memory and Intel Core2 Duo CPU.

<table>
<thead>
<tr>
<th>Method</th>
<th>Personalized</th>
<th>k-means (all)</th>
<th>EM (top 2)</th>
<th>EM(m=20%)(top 2)</th>
<th>Efficient EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time(ms)</td>
<td>225.06</td>
<td>0.17</td>
<td>28.15</td>
<td>10.58</td>
<td>0.42</td>
</tr>
</tbody>
</table>

We discovered that k-means performed the best for a typical user (0.17 ms). Since this method recommends the same items to users in the same cluster, it does not need to compute each user-URL pair’s similarity, but only each cluster-URL pair’s similarity, thus reducing much computing time. If we compare the performance of EM (top 2) which was 10.58 ms with EM (all) which was 20.15 ms, the performance of EM (top 2) was almost 2 times faster than EM (all). It shows that decreasing the number of clusters can significantly improve the recommendation efficiency without sacrificing accuracy. Comparing the performance of EM (top 2) which was 10.58 ms with EM (m= 20%, top 2) which was 0.42 ms, the performance of EM (m= 20%, top 2) was nearly 25 times faster than EM (top 2). This shows that if we properly select the number of recommendation URLs | R |, the recommendation efficiency can significantly improve without losing the accuracy. Based on efficient EM’s performance, we can conclude that both reducing number of clusters and reducing number of URLs are important and dependent.

We discovered that our approach was very efficient (0.37 ms) and improved over general EM methods. Despite that our method was nearly 2 times slower than k-means (0.37 ms vs. 0.17 ms), we were however significantly more efficient than other methods. Even though k-means was the most efficient, it had the worst average rank, novelty and diversity. Thus, by combining content quality and performance, we conclude that our method was the best.

V. CONCLUSION

Personalized recommendation provides accurate results, but is time consuming particularly when having large user base with high engagement. CF and content-based recommendations suffer from scalability, cold start, and possible poor accuracy. Clustering-based recommendations improve recommendation efficiency, but accuracy suffers and results may not be personalized. Other issues include selecting the clusters and items in each cluster to optimize the recommendation efficiency.

We presented our method called efficient EM-based recommendation for personalized results, while greatly improving recommendation efficiency. Our method performed better than personalized, k-means and EM-based methods; after taking into account average rank, diversity, novelty and recommendation time. The result showed that our method was better than the existing clusters’ recommendations. In addition, our method can be applied to every content-based personalized recommendation, such as news recommendation, book recommendation, and application recommendation. For future work, we plan to test the viability of our method on other browsing and product datasets. We also plan to evaluate our method with CF and content-based recommendations to provide a more comprehensive evaluation. Finally, we want to explore the selection process for clusters and items to further improve the accuracy of recommendations.

VI. ACKNOWLEDGEMENTS

This research was supported by the grants from the National High Technology Research and Development Program of China (Grant No.2014AA015203), and the Fundamental Research Funds for the Central Universities of China (Grant No. WK0110000042).

REFERENCES


