Identifying Popular Items by Analyzing Randomized Algorithms

[#1]Rambabu Pemula, [*2] Kumar Jetti, [*3]Srinivasa Rao Divvela, [*4]Devarapalli Anandam

[#1]*Department of CSE, Nimra Institute of Engineering & Technology, Ongole, Andhra Pradesh,India.*
[*2]*Department of CSE, CMR College of Engineering, Hyderabad , Andhra Pradesh, India.*
[*3]*Department of CSE, SSN College of Engineering, Ongole, Andhra Pradesh, India.*
[*4]*Department of CSE, Nimra Institute of Engineering & Technology, Ongole, Andhra Pradesh,India.*

rpemula@gmail.com [1]
kumarkanna.j@gmail.com[2]
anandbabu.1361@gmail.com[3]
anandbabu.1361@gmail.com [4]

***Abstract*— Interactive computational systems are helping people to maintain social information; these systems are known as social navigation systems. These help the user in performance guiding and making the decisions in selecting the data. Suggestion of popular items and ranking can be performed depending on the individual feedback, where the individual feedback is obtained by displaying group of suggested items. Selection of items is based on the individual or users demand the objective is to propose true popular items and quickly studying the true popularity ranking of items. The difficulty is that making suggestions to users can reinforce popularity of some items and distort the resulting item ranking. Many applications like Tag suggestions and search Query suggestions have been affected by the problem of ranking and suggesting items .In this paper we present the algorithms like naïve,PROP,M2S,FM2S for ranking and suggesting popular items.**

***Keywords*— Popular Items, Ranking, Suggestion, Randomized Algorithms, itemized sets.**

I. INTRODUCTION

The objective is to quickly determine the factual attractiveness ranking of items and to suggest the popular items. The Items can be suggested to users in order to simplify the tasks like browsing or tagging of the content. A precise application of tagging the content where items are "tags" employed by the users to content such as photos (e.g., Flickr), videos (e.g., YouTube), or Web pages (e.g., del.icio.us) for their future recovery or personal information management. The general idea of social tagging is that the user can select any set of tags for an information object according to their preference. many social tag applications are provided with tag suggestions that are made based on the history of tag selections by the users . Fig. 1 shows an example user interface to enter tags for a Web page.

This paper proposes the algorithms and analyzes the performance of suggesting the popular items to users in a way that enables learning of the user's true preference over items. The true preference is the preference upon items that would be perceived from the user's selections over items without exposure to any suggestions.

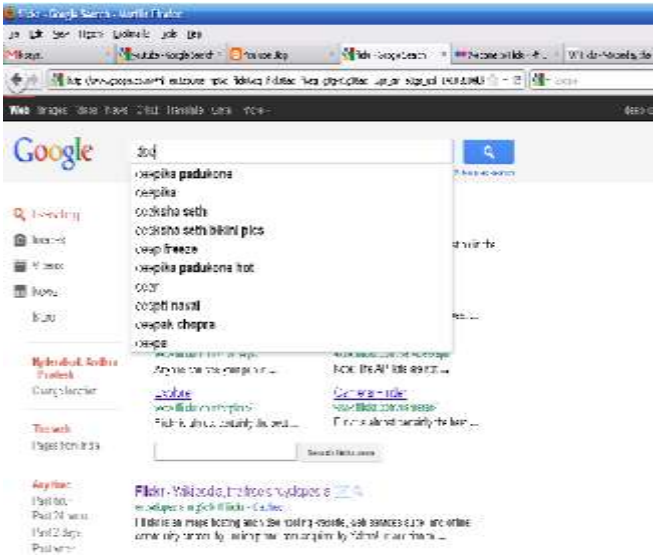

Fig. 1: User interface to enter Tags

A simplified structure for ranking and suggesting popular items (that appears in common use in practice) presents a fixed number of the most popular items as obtained from the previous selection of items. The analysis shown here suggests that simple scheme can lock down to a set of items that are not the true most popular items if the popularity bias is large, and may obscure determining the true preference over items. In this paper, we present the substitute algorithms that are designed to avoid cavalries and to provide performance analysis of the ranking limit points and popularity of the suggested items. The formal analysis is out of the scope of this paper if it is concerned with the speed of algorithms.

II. RELATED WORK

We present a more specific description of the problem of ranking and suggesting items that we consider, and define a user's choice model that we use for our analysis. let us consider how the user select's items. User selects an item from the entire set of items by sampling, using the true item popularity distribution *r*. Where

$$r = (r_1, r_2, \ldots, r_c)$$

be the Users' true preference over the set of items C and r called true popularity rank **scores.** Otherwise, user does the same but confines his choice to items in the suggest set.

Pr{selected item =i|suggestion set=S}=(1-p_S)ri+ps r_iiЄS/∑jЄS r_j

The naive algorithm which suggests a fixed number of the popular items, fails to determine the true popularity ranking of items if the imitation probability in the user's choice model is adequately large.This can be explained in the pseudo code of Fig 2.

TOP (TOP POPULAR)
Init: $V_i = 0$ *for each item* i
At the t-th item selection:
  If *item* i *selected:*
    $V_i \leftarrow V_i + 1$
  $S \leftarrow$ *a set of* s *items with largest* V *counts*

Fig.2: Pseudo code of Navie Algorithm implementing TOP

The item's rank score which is equal to the number of selections of this item in the past can be maintained by using the ranking rule. For the naïve algorithm and the algorithms that are implemented in future, we initialize VI=0 for each item. The implicit assumption is that we assume no prior information about the popularity of items, and hence, assume that all items are equally popular. The suggestion rule sets the suggestion set to a set of the top s most popular items with respect to the current popularity rank scores.

III. PROPOSED ALGORITHMS

We define two ranking rules called rank rule 1 and rank rule 2 and various t suggestion rules like

a) **Frequency Proportional**
b) **Move to Set**
c) **Frequency Move to Set**

Ranking Rules

Ranking rules are classified into rank rule1 and rank rule 2.

- *Rank rule 1*

A simple ranking rule is the one that we already encountered in the algorithm TOP, where the rank score for an item i is incremented by 1 whenever a user selects this item. Its Pseudo code is proposed in Fig 3.

Init: $V_i = 0$ *for each item* i
At the t-th item selection:
  If *item* i *selected:*
    $V_i \leftarrow V_i + 1$
    $\rho_i \leftarrow V_i/t$

Fig.3 Pseudo Code for Ranking Rule 1

- *Rank rule 2*

We noted that rank rule 1 may fail to discover the ranking order of the true popularity if used with suggestion rules such as TOP. Here, rank score updated only for an item that was not suggested. Slow rate of convergence. Its Pseudo code is proposed in Fig .4.

Init: $T_i = 0$, $V_i = 0$, *for each item* i
At the t-th item selection:
  For each item i*:*
    If *item* i ***not*** *suggested:*
      $T_i \leftarrow T_i + 1$
    If *item* i *selected:*
      $V_i \leftarrow V_i + 1$
    $\rho_i \leftarrow V_i/T_i$.

Fig. 4: Pseudo Code for Ranking Rule 2

A. *Frequency Proportional*

PROP is a randomized algorithm that for each user presents a suggestion set of items, sampled with probability proportional to the sum of the current rank scores of items. We will later show analysis that this suggestion rule combined with rank rule 1 is more robust to imitation than TOP, but there still may exist cases when it fails to learn the true popularity of items. Note: also that the algorithm is computationally demanding when the number of items and suggestion set size s are non small; it requires sampling on a set of elements as pseudo code proposed in Fig.5

PROP (FREQUENCY PROPORTIONAL)
At the t-th item selection:
  Sample a set S *of* s *items with probability* $\propto \sum_{j \in S} \rho_j$

Fig. 5 Pseudo Code for PROP

B. Move to Set

M2S is a random iterative update rule of the suggestion set of items ,where the suggestion set is updated only when a user selects an item that is not in the suggestion set presented to the user. M2S suggests the last used item for the suggestion set size of one item which is a recommendation rule used by many user interface designs. Due to the random eviction of items from the suggestion set, M2S is different from suggesting the last distinct used items for the suggestion set size greater than one item although the rule prefers recently used items. We will show how exactly this update rule tends to bias the sampling of the suggestion set with respect to true popularity rank scores of items. As an aside, note that M2S relates to the self-organized sorting of items known as move-to-front heuristic as proposed in pseudo code in Fig.6.

M2S (MOVE-TO-SET)
At the t-th item selection with item i selected:
  If item i **not** *in the suggestion set S*
    Remove a random item from S
    Add i to S

Fig. 6 Pseudo Code for M2S

C. Frequency Move to Set

For each item, the algorithm keeps a counter of how many users selected this item over users that were not suggested this item. The rationale is not to update the counter for items that were suggested and selected by users in order to mitigate the positive reinforcement due to exposure in the suggestion set. Furthermore, a selected item that was not suggested does not immediately qualify for entry in the suggestion set (as with M2S), but only if its counter exceeds that of an item that is already in the suggestion set as in Fig 7. In addition, specific to FM2S is that the eviction of an item from the suggestion set is over a subset of items with smallest counter.

FM2S (FREQUENCY MOVE-TO-SET)
Init: $W_i = 0$ *for each item i*
At the t-th item selection with item i selected:
  If item i **not** *in the suggestion set S*
    $W_i \leftarrow W_i + 1$
    $E = \{j \in S : W_j < W_i\}$
    If E is nonempty
      Remove a random item from S that is in E
      Add i to S

Fig. 7: Pseudo Code for FM2S

M2S and FM2S learn true popularity ranking that are lightweight. Self-tuning in that they do not require any special configuration parameters.FM2S confines to displaying only sufficiently popular items as the suggestion set can displayed as shown in the Fig. 8.

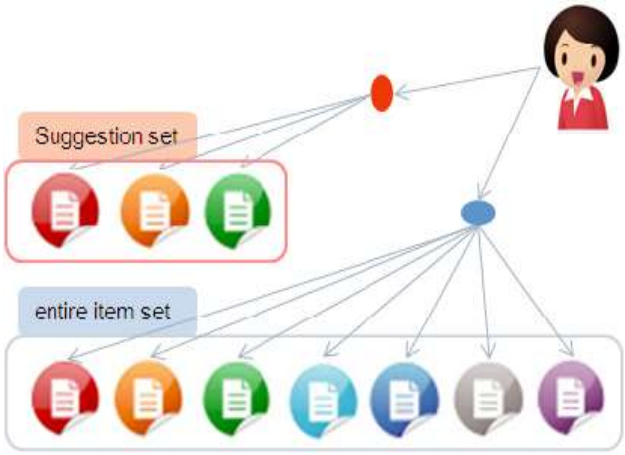

Fig 8: The Proposed Suggestion Set to the user

IV. SYSTEM DESIGN & IMPLEMENTATION

In this paper the proposed and studied algorithms like naïve, PROP, M2S, FM2S algorithms were used to design a framework for ranking and suggesting popular items for the users. Initially to add item sets into the framework we used the pseudo code in Fig.9 .And when the user wants to view the item sets a pseudo code is proposed in Fig. 10. And when we want to update the popular item sets a pseudo code is proposed in Fig. 11.And when the item sets those are not utilized can be deleted using the pseudo code is proposed in Fig. 12.

```
CoreHash
aCoreHash = new
CoreHash();
Category aCategory
= new Category();
aCoreHash = new
CategoryDAO().listC
ategories();
Enumeration enu =
aCoreHash.elements
();
while(enu.hasMoreEl
```

Fig. 9: Code to add new items

```
String header = request.getParameter("header");
String role = (String) session.getAttribute("role");
CoreList aCoreList = new CoreList();
CoreHash cCoreHash = new CoreHash();
try{
int categoryid = 0;
int itemid = 0;
ItemDAO itemdao = new ItemDAO();
CategoryDAO categorydao = new CategoryDAO();
aCoreList = itemdao.listItems();
cCoreHash = categorydao.listCategoryNames();
if(!aCoreList.isEmpty())
{
if(header.equals("list") && role.equals("admin"))
{
%>
<td width="18"><div align="center">
<input type="checkbox" name="ch" id="checkbox"
onclick="SetChecked('ch')"/>
</div></td><%
}
```

Fig. 10: Code to view items

```
try
{
int categoryid =
Integer.parseInt(request.getParameter("cate
goryid"));
int itemid =
Integer.parseInt(request.getParameter("ite
mid"));
String itemname =
request.getParameter("itemname");
String desc = request.getParameter("desc");
Item aItem = new Item();
aItem.setCategoryID(categoryid);
aItem.setItemID(itemid);
aItem.setItemName(itemname);
aItem.setItemDesc(desc);
ItemDAO aItemDAO = new ItemDAO();
boolean flag = aItemDAO.updateItem(aItem);
}
catch(Exception e)
{
LoggerManager.writeLogWarning(e);
}
```

Fig. 11: Code to update items

```
String
target="ViewItems.jsp?header=list";
try{
ItemDAO aItemdao = new ItemDAO();
String ch[] =
request.getParameterValues("ch");
for(int i=1;i<ch.length;i++)
{
aItemdao.deleteItem(Integer.parseInt(
ch[i]));
}
}
catch(Exception e)
{
LoggerManager.writeLogWarning(e)
;
}
RequestDispatcher rd =
request.getRequestDispatcher(target)
;
rd.forward(request,response);
```

Fig. 12: Code to delete items

V. RESULTS

The following are the screen shots of the paper

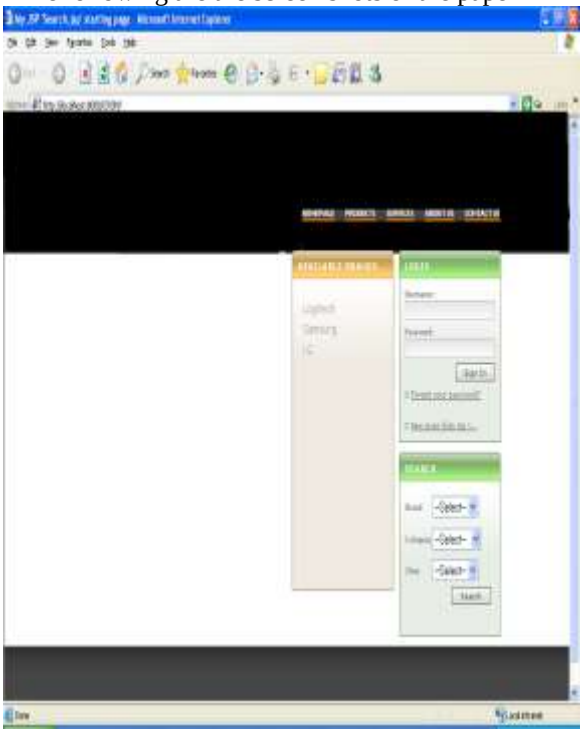

Fig. 13: Login page

Fig.14: Items added to cart

Fig. 15: updated cart items

Fig. 16: suggested and available items sets

VI. CONCLUSION

In this paper , to suggest the most popular items based on ranking and popularity of items we analyzed and proposed the randomized algorithms like naive, PROP, M2S, FM2S.We assessed quality of suggestions which are measured by true popularity of suggested items, and we identified how limit ranking of items are related to true popularity ranking. By learning the true popularity ranking of items, the proposed objective of suggesting true popular items can be quickly achieved by using the proposed algorithms.

REFERENCES

[1] Dwayne Bowman"Identifying the items most relevant to a current query based on items".

[2] S. Sen, S.K. Lam, A.-M. Rashid, D. Cosley, D. Frankowski,J. Osterhouse, F.M. Harper, and J. Riedl, "Tagging, Communities,Vocabulary, Evolution," Proc. 2006 20th Anniversary Conf. Computer Supported CooperativeWork (CSCW), 2006.

[3] A.L. Baraba´si and R. Albert,"Emergence of Scaling in Random Networks," Science, vol. 286, pp. 509-512, 1999.

[4] S. Brams and P. Fishburn, Approval Voting. Birkhauser, 1983.

[5] S. Chakrabarti, A. Frieze, and J.Vera, "The Influence of Search Engines on Preferential Attachment," Proc. Symp. Discrete Algorithms (SODA), 2005.

[6] J. Cho, S. Roy, and R.E. Adams, "Page Quality: In Search of an Unbiased Web Ranking," Proc. ACM SIGMOD '05, 2005.

[7] S. Golder and B.A. Huberman, "The Structure of CollaborativeTagging Systems," J. Information Science, vol. 32, no. 2, pp. 198-208,2006.

[8] Linden, G.; Smith, B.; York, J.; "Amazon.com recommendations: item-to-item collaborative filtering",Internet Computing, IEEE ,Volume: 7 Issue:1-On page(s): 76 - 80
[9] Junqiang Liu,Yunhe Pan"Mining frequent item sets by opportunistic projection",ISBN:1-58113-567-X doi>10.1145/775047.775081
[10] Shilad Sen,Jesse Vig,John Riedl,"Tagommenders: connecting users to items through tags",WWW '09 Proceedings of the 18th international conference on World wide web.