RECOMMENDATIONS:

In this application we made a recommendations system based on tags. User, while registering an item for auction or sell, chooses appropriate tags for the item from the list of tags displayed to him. This selection can be single tag or multiple tags.

TRENDING PART:

Algorithm:

The basic idea behind the algorithm for trending part is as follows. We have in our database a table for recommendations storing tag values against each user. The table has three columns one for user name, one for tag values and the last one for recommended items. Tag values column has the string in the format “tag1=tag1_value, tag2=tag2_value, etc….”. Here we sum up the values of tags for all users and sort the tags in the descending order of their tag values. Based on the tag values we take the top 5 tags and show the user, items with maximum number of matching with these tags.

Implementation in terms of code:

Create an array with keys equal to tags (all tags) and initialize the values for all the keys to zero. Now we to go through all the users in the recommendations table and add the tag value of a particular tag present against user to the array we initialized in the beginning. We do the same for all the users and towards the end we get the array filled with the summed up values. We sort this array using “arsort” function. Then select top 5 tags. Now with these tags in hand, we move to items table and for every item we give a count value based on number of tags that item has which match to the top 5 tags. Based on this count value we recommend items to new user.

ORDER BASED RECOMMENDATIONS:

Algorithm:

In the recommendations table corresponding to every user, we have tag values. To optimize the database we just store the tags that have non-zero values for that user. We make values of all the tags to zero before we start our procedure. Now based on his order history we add a specific value to the particular tag in the tag_values in the table for recommendations if that tag is already present or else we introduce this tag to the tag values column. And push back this string to the tag values column.

Implementation in terms of code:

We begin with mysql query to select tag values from recommendations table for the user("SELECT tag_values from rec_pre where user_nm = user name). Taken them into an array named $a. Then make all the tag values to zero using for loop. Now we extract the item_ids from the orders table for this user and now looping through each item_id, we push tags corresponding to this item to an

array. Now we loop through this new array and check a value against tag values in array $a and increase the value if tag is already present or add it into array if not.

SEARCH BASED RECOMMENDATIONS:

Algorithm:

While displaying the recommendations in the profile page for a particular user, we take into consideration the search history of the user. This is accomplished by going through all the searches of that user and correspondingly adds the value to his tag values, which in turn has effect on his recommendations.

WATCHLIST BASED RECOMMENDATIONS:

Algorithm:

Similar to order and search based recommendations. We extract the item_ids for that user from data base table. Get the tags for that item and increase the values for tag values for tat corresponding user.

GENERAL:

Recommendations part works by going through the order history first (as it is given highest priority), then watchlist and at the end search history. At the end of the code we insert the recommended items into database table rec_pre under the column rec_items. Using this column we directly get access to the items recommended for the user.

FOR DEVELOPERS:

We used global variables for the preference added to tag values to order, search history and watchlist which enable developers to just change this value and not the whole code. We have implemented recommendations based on tags alone which can be extended to further recommendations.