DATA MINING FOR USER DATA ANALYSIS ON ONLINE PROGRAMMING JUDGES.

Mini Project on Data Mining.

Submitted by -

106109028 Gopal Rander

106109041 Kunal Kapadia

106109059 Parth K. Dandiwala

106109060. Patel Vaibhav D.

B. Tech. VIII Semester,

Date – 22nd April, 2013

Abstract

In this project, we do a detailed analysis of a user and the problem set on an online judge.

Generally, following sequence of events happen when a user submits solution for a problem on online judge.

1. User submits the code in certain language.
2. The online judge compiles the code, test it on some input files, and verify whether the submission is wrong, Time limit Exceeded, or a correctly accepted one.

When a user solves a new problem, Based on these three parameters, his points are calculated

1. Frequency at which the problem is solved.
2. Number of users who have solved the problem.
3. Rating of the users who have solved a particular problem.

Other than that, for every user in the database, we have a Performance Statics containing

1. Number of correct and wrong submissions.
2. Rating v/s week graph.
3. List of problems solved.
4. Highest Rating ever achieved. And many more…

Using above measures, the following utilities will also be available

1. Best 5 coders of week/ month.
2. Next 6 recommended problem for every user.
3. Comparing user profiles.

Implementation: We are implementing this application tool in C++, using file storage as database.

Structures and Constants Used

1. Problem - Data items associated with a single problem.
    1. Problem Name.
    2. Problem ID. / Index.
    3. Tag List.
    4. Rating of the problem.
    5. Time added.
    6. Attempted count, Solved AC count, User solved count.
    7. List of users solved this problem
    8. Last time problem was solved.
    9. Frequency at which the problem is solved.

2. User - Data items associated with a single problem.
    1. User Name.
    2. User ID. / Index.
    3. User Rating.
    4. Highest Rating.
    5. List of solved problems.
    6. No. of Submissions and No. of AC.
    7. Join time and last activity time of user.
    8. Pointer to first problem solved in last 1 month time.

Functions with description

1. void prec (void)

Load the problem data and user data from secondary storage (files) to main memory (program). Loaded in problems vector and users vector to be processed/ analyzed.

2. void postc (void)

Write back the problem DB and user DB after a sequence of events like addition of problem or user, and submissions by users. Any changes done are written at the end to storage files or if explicitly mentioned. Intermediate changes are stored in Main memory only.

3. void add_problem (void)

For adding a new problem in DB. Requires name and tag list from the user. Start time is set to the maximum of Last time of all the users. Default rating of each problem is 3.00.

4. void add_user (void)

For adding a new user in DB. Requires name of the user. Start time of the new user is maximum of all the last time of already existing users and existing problems. Default rating of the user is 0.00, which changes according to the submissions.

5. bool copy_check (problem_id, user_id)

Checks if user[user_id] has already solved the problem[problem_id]. If yes, return true. Else false.

6. double normalize (value)

Normalize a value in Range [0, 5000] to range [0.75, 1.75]. (Used as multiplication factor)

7. double aggr (vector u_list)

Calculates the aggregate value of all items in u_list using following formula.

8. double cal_rating (p_id, u_id, time)

Calculates the new rating of problem[p_id] when a user[u_id] submits a solution. Factors taken into account are as follows.

a. Rating of the users who have already solved that problem[p_id].

b. Number of submissions for problem[p_id].

c. Number of correct submissions for problem[p_id].

d. Number of users who have already solved the problem[p_id].

e. frequency at which problem[p_id] is solved.

We have formulated the following formula to calculate the expanded multiplying factor.

y =

Then we normalize it in range [0.75,1.75]

9. void query (void)

Asks for the user_id, problem_id and time for particular submission. Process this data and calculates the new rating for problem[p_id]. Changes are propagated in all the users' ratings who have solved the problem[p_id]. Each time, submission is checked if it is a copy submission. Problem data and User data are updated accordingly.

---

10. void a_tools (void)

Provides analysis tools for judge like

1) Top 6 recommended problems for a particular user

2) Top 5 coders of last month.

3) Compare two coder's performance statistics.

'set' used in the next 3 recommendation functions is the set of problem ID's, which are already solved by user or is being recommended by some other recommendation system.

Following are 3 recommendation systems.

11. void rec1 (u_id , set , p1, p2)

This is tag based recommendation system.

- List of all tags of all the problems solved by user[u_id] is generated and sorted on the basis of count of each tag.

- For each problem not in 'set' , number of matching level of tag list with the above list are noted.

- Two problems with maximum level matching are recommended as p1 and p2.

12. void rec1 (u_id , set , p1, p2)

This is user rating based recommendation system.

- For each problem not in 'set', aggregate value for list of users who have solved the problem, is calculated.

- These aggregate values are plotted on a line along with the rating of user[u_id].

- The nearest two neighbors are returned as p1 and p2.

13. void rec3 (u_id , set , p1, p2)

This system is based on the rating of users and problems solved by those users.

- 2*k nearest rated users of user[u_id] are considered. (k constant defined)

- A list of problems which are not in 'set', solved by those users is generated and sorted on the basis of count of users.

- Top two problems with maximum count are recommended as p1 and p2.

14. void get_top_from_last (res)

Populates the vector res with the top 5 users of the current month. Uses the left pointer of each user and calculates the change in rating during that time. The one with largest change is on top.

15. void compare_users (u_id1, uid2)

Analysis tool to compare the user stats listed below.

1. Rating.
2. Highest Rating ever achieved.
3. Number of problems solved.
4. Accuracy.
5. Joining time.
6. Last activity time.
7. Frequency of problem solving.
8. Tags associated with problems being solved

(in decreasing order of count)

16. void diplay_ratings (void)

Display rating of all users in sorted order.

Main Program

The main Program provides a menu with following options ->

1: Add a problem

2: Add a User

3: Make a problem solving attempt

4: Use analysis tools

5: Well, I would like to check your data

6: Display the ratings of all users

7: Done. I would like to go offline