Project Report

Adaptive Question Recommendation System

By – Mihir Ahlawat
Data Science Intern
1st June 2019 - 25th July 2019

Acknowledgement

I, Mihir Ahlawat would like to convey my gratitude to Mr. Sujit Bhattacharyya , Chief Digital Officer of Career Launcher, New Delhi for emphasizing on the 2-months Summer Internship Program and giving me the platform to interact with industry professionals and helping me immensely during the course of the project. I extend my warm regards to everyone who helped me during my internship.

INDEX

Introduction

Adaptive question recommender system aims at providing students who are operating on a test at the moment with a set of questions depending on whether a student attempts the current question correctly or not. If a student attempts a question correctly, a more difficult question is posed to him, on the other hand if a student attempts that question incorrectly, similar set of questions are then recommended to him to solve. This model was trained on dataset which contained questions from the module “Percentage” which contained 189 unique questions.

Dataset

Two Datasets were used for analysis and training -“users” and “percentages”. Users dataset had attributes consisting of ‘user_id’, ‘section_id’, ‘avg_per’, ‘top_3_avg_perc’, ‘best_per’, ‘nproc’. The main motive of the dataset was to classify users on the basis of their academic abilities.

Percentage dataset contained attributes such as module name, question id, student id, level of difficulty of the question, time taken by the student to attempt the question, whether the student skipped the question or not and if attempted, whether he/she attempted it correctly or not. This dataset was aimed at providing insights about each attempt of user on various questions and the attributes related to that attempt.

Exploratory Data Analysis

For every unique question we needed to find attributes which recognised them uniquely. Through observation we found out that time taken by a user, attempted percentage by all users for that question (ratio of number of users who spent time on that question and marked an option to number of all users who saw that question and spent some time on it), accuracy of that question (ratio of users attempting that question correctly to number of users marking that question) could be regarded as distinctive features for a question.

Our Analysis started with exploring the various facets of percentage dataset. Firstly the number of unique questions and the number of unique users were found out for percentage dataset which came out to be 189 and 1410 respectively.

Next frequency vs time plot was analysed for the complete Percentage dataset. We found out that most of the questions were attempted between the time frame of 0-120 seconds.
After realising the mode of occurrence we processed the dataset by setting marked option 'x' as not attempted and grouping the dataset on unique questions. This step was aimed at analysing the dataset at the question level where we can find the attributes with respect to each individual question. After grouping the dataset on questions, we aggregated the groups by mean. This provided us with accuracy and attempted percentage of each question along with the mean time taken for that question to be attempted. We then tried to visualise the pattern by plotting each question on accuracy vs time raken graph and saw the general trend of increasing time with decreasing question's accuracy.

To get a better idea of attributes regarding unique question we tried calculating of attribute with respect to different classes of users attempting that question. The classes were defined on the basis of top 3 average percentile of users from 'users' dataset.
Two classes were defined - Above 90 percentile users and Between 70 and 90 percentile users. Same grouping and aggregation was performed on groups mapped

on to these different users classes. We found out that the accuracy of a question shot up by a certain percentage for above 90 percentile user class and the time taken was also somewhat less by a factor as compared to between 70 and 90 percentile user class.

We decided to take attributes corresponding to above 90 percentile user class as distinguishing attributes to consider in our model as it had definitive and reliable values for each question.

The only problem was to find a reliable factor of time that could serve as a distinguishing factor for a question. As our dataset contained flukes, vague attempts or stuck up attempts, we needed to find a distinguishing time which could uniquely identify a question and is free from all this noise.

We analysed time on various dimensions. First we considered simple mean value of time, then moved onto adjusted time which was just mean time considered for the interquartile range (between 25 percentile and 75 percentile), then moved mean time for data points in which question was attempted (that is an option was marked), then took the same time in the interquartile range, then repeated the same processing for not attempted data points. This was followed by repeating the same process for correctly marked attempts and incorrectly marked attempts.

We plotted these different times for a random question and argued as to which time optimally describes a question. We found that adjusted correct mean time best describes a question’s attribute. So we took as a distinctive feature of time rather than plain average.

After all this analysis we clustered the question set into 10 different clusters. Each cluster defines a set with a specific ‘difficulty’ level and we can argue that each question in a cluster is similar to one another.

Correlation between users

Clustering questions into different levels solved our problem of recommending similar or different question to user after an attempt. But do we need to recommend all the questions?
There may be users who find a comparatively difficult question easier or an easy question difficult. Here's where user based collaborative filtering comes into play.User based collaborative filtering recommends items by finding similar users to the active user.

To find users similar to a user we take a factor on the basis of which we will distinguish one user from another. Here that factor is time taken to attempt a specific question. Time taken to correctly attempt a question, to incorrectly attempt a question and just to spend some time and not attempting that question altogether should be used distinctively to find similar users.

Therefore we scale time by a factor by 1000 if a person correctly attempts a question (awarding him points for correctly attempting that question), scale time by -1000 if a person incorrectly attempts that question (punishing him for incorrectly attempting that question) and scale time by negative time taken if he spends time and skips the questions (punishing him for not attempting but not that severely).
Now we got a dataframe which points denoting question attempted by a user with scaled time. We make a pivot table, which is all questions as rows and all users as columns. For a question attempted by that user, the value is filled with time otherwise it's filled with nan value.

We now find a correlation between users based by all the common questions they have attempted and the time taken to attempt that question (Note that the time taken here is the scaled value of original adjusted correctly attempted time of the question).

Correlation quantifies the degree to which a relationship
between two variables can be described by a line.

The original formula for correlation, developed by Pearson
himself, uses raw data and the means of two variables, X and Y:

$$\rho_{X,\,Y} = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}}$$

In this formulation, raw observations are centered by subtracting their means and re-scaled by a measure of standard deviations.
This function was implemented by df.corr() function in pandas library which found out the correlation between different users on the common set of questions attempted by them. Minimum period of 10 questions was set, which set minimum of 10 questions will be regarded to compute the correlation between two users.After performing df.corr() on our questions vs users pivot table, we get user vs user correlation table, which had correlation between users as their values. We can easily find out the similar users to our current user by sorting the row pertaining to our current user in descending fashion.

Building the model

We already got our similar questions set and similar user groups.
To build our model we just need to recommend similar/different questions to our current user based on whether he attempted the current question correctly or not.
To recommend questions we find out top 5 users which are similar to our current user. We find out the questions attempted by them and not by our current user. These are the questions we are going to recommend our current user, Let’s call these questions ‘q’. We will segregate questions in q on the basis of which cluster they belong to. We need to define a parameter so that we can access the difficulty of a cluster. We find out the data points (questions) closest to each cluster centres. We define the ‘difficulty’ of these clusters on the basis of accuracy of these data points (questions). We will start now from the least difficult cluster. We will recommend a question from that cluster. If our user attempts that question correctly, we will recommend him a question from a more difficult cluster. If he attempts it incorrectly, we will recommend him a question from the same cluster. (Note that difficulty of clusters are assigned according to the accuracy of it’s mediod and all the questions here are from set q). We have developed a function along with our main recommending function to compare the attributes of current attempting question and previously attempted question to give a better visualisation of working of our recommender system.

Conclusion

The model developed recommends question for a user to attempt on the basis of whether he attempts his currently attempting question correctly or correctly. This model was built on ‘percentage’ module. This model was built on 189 questions and 1410 distinct users.