Recommending Personalized Review Questions using Collaborative Filtering

Zain Kazmi
University of Toronto
Mississauga
zain.kazmi@utoronto.ca

Wafiqah Raisa
University of Toronto
Mississauga
wafiqah.raisa@mail.utoronto.ca

Harsh Jhunjhunwala
University of Toronto
Mississauga
harsh.jhunjhunwala@mail.utoronto.ca

Lisa Zhang
University of Toronto
Mississauga
lczhang@cs.toronto.edu

ABSTRACT

This paper presents the work in progress towards a tool where CS1 students receive personalized review questions to prepare for their term tests. Specifically, the tool recommends multiple choice and coding questions. The recommendations are generated using collaborative filtering, based on students' past performance on these questions. We test recommendation engine models based on last year's student data, and present offline experiments that show the promise of this approach.

Author Keywords

CS1, Collaborative Filtering, Machine learning, Recommendation Engine

INTRODUCTION

With increasingly large class sizes in computer science courses, it is difficult for students to receive personalized feedback on how to study and improve. We aim to help our CS1 students revise for their term tests by providing each student with a set of personalized review questions.

The CS1 course at our institution uses an educational computer programming platform called PCRS [3] to deliver some course content. PCRS pairs video-based instruction with relevant pre-lecture and post-lecture exercise questions. Our goal is to re-purpose and recommend some of these questions for students to review, and integrate the review tool into the PCRS platform.

As a proof-of-concept, we build collaborative filtering models that predict the questions that will likely challenge students. We use historical data consisting of CS1 student attempt history to build these models. We hope to test the effectiveness of a recommendation engine approach in an actual CS1 course environment.

Week	Question Type	Num Students	Num Problems	Percent Students Solved	Avg Attempts Until Solved
1-4	MC	1,055	128	99.7%	2.44
1-4	Code	1,044	41	98.4%	2.89
5-8	MC	1,029	87	99.5%	3.48
5-8	Code	978	20	96.5%	3.84
9-12	MC	961	98	99.5%	3.33
9-12	Code	845	18	96.6%	3.87
All	MC	-	313	99.7%	2.78
All	Code	-	79	97.9%	3.23

Table 1. Data Submission Distribution

BACKGROUND AND RELATED WORK

Although our task of recommending previously-seen content for test revision differs from helping students navigate new content [1], we take some inspiration from the recommendation system approach of [5]. Additionally, EduRank [4] uses a collaborative filtering approach that predicts student performance over study questions using the historical student performance.

PCRS DATA

The CS1 course at our institution has between 800-1100 students. In each of the 12 weekly modules, students use PCRS to watch a set of videos, and complete a set of multiple choice, short-answer, and coding questions. These questions are graded for correctness, and students can re-attempt a question any number of times. In fact, 94% of students re-attempt questions before term tests. PCRS collects interaction and performance data. In particular, the platform logs student attempts to each question. We re-purpose the historical record from a past term to evaluate collaborative filtering approaches to review question recommendation.

Sixth SPLICE Workshop at L@S 2020 "Building an Infrastructure for Computer Science Education Research and Practice at Scale", SPLICE'20, August 12, 2020, Virtual Event

Model	FCP	MAE	Hardest5
Random	0.49	2.39	33%
Baseline	0.76	1.35	60%
KNN	0.76	**1.23**	60%
SVD (dim=50, epochs=20)	**0.77**	1.27	**61%**
SVD++ (dim=50, epochs=1)	0.75	1.35	60%
Test Results			
SVD (dim=50, epochs=20)	0.76	1.26	58%
KNN	0.76	1.24	55%

Table 2. Collaborative Filtering Performance

The historical data contains 313 multiple choice questions and 79 coding questions with at least one student submission. It also contains submissions from 1,057 unique students, including 896,031 submissions to multiple choice questions and 244,384 submissions to code questions. We randomly split the students into 846 students in the training and validation sets, and 211 students in the test set.

Table 1 summarizes our data. We split the course into three portions, since there are major term tests after weeks 4, 8, and 12. Note that there are fewer but longer questions in later weeks, and students require more attempts on average to solve later questions. Still, a vast majority of the students eventually solve these questions.

COLLABORATIVE FILTERING MODELS

To build the recommendation models, we compute the number of times that each student attempts each question until they obtained full marks. This value is a proxy for the student perceived difficulty of the question: the higher the attempt, the more challenging the question. Since we wish to recommend questions that a student will find challenging, we treat the *number of attempts until correct* as the student's *rating* of the question. In other words, an element of the rating matrix represents the number of times a student attempted a question until solving the question, with the maximum allowed attempt set to 15. For the small number of students who attempted but did not solve a question, we assign the max rating of 15.

We use the Python library `surprise` [2] to build the collaborative filtering models. We experiment with a baseline model that takes into account the average difficulty of a question, a nearest-neighbour approach, and singular value decomposition approaches.

COLLABORATIVE FILTERING RESULTS

We use 5-fold cross-validation to test each model, and use the held-out test set to compute the test statistics for the best models. In our validation set, we randomly select and remove 15 ratings from each student, and use the model to predict those 15 ratings. We use 3 different metrics to measure the performance. The Fraction of concordant pairs (FCP) and the Mean Average Error (MAE) are standard metrics computed using the library `surprise`. Furthermore, for each row in our validation set, we predict the "Hardest 5" of the 15 removed ratings, and compute the accuracy compared to the ground-truth. This metric is important because our primary goal is to recommend problems, so the order of the predictions matter more than the predicted ratings.

Model	Metric	Fold1	Fold2	Fold3	Fold4	Fold5
Baseline	FCP	0.763	0.763	0.766	0.759	0.760
SVD	FCP	0.766	0.771	0.772	0.758	0.769
Baseline	MAE	1.372	1.384	1.328	1.333	1.326
SVD	MAE	1.296	1.296	1.242	1.268	1.248
Baseline	Hard5	60.2%	59.3%	60.7%	60.0%	60.0%
SVD	Hard5	60.4%	61.5%	61.1%	60.4%	60.8%

Table 3. Comparison of SVD and Baseline over 5 folds

We add two baselines to help interpret the performance metrics. The "random" baseline assigns a prediction from a normal distribution centered on the average ratings for the student and the question. The "baseline" model is similar but does not introduce randomness.

Table 2 shows that SVD model performs marginally better than other models when predicting user ratings. The baseline is strong, since the variation in question difficulty explains much of the data. Still, Table 3 shows that the SVD model over-performs the baseline in every one of the 5 folds, so the difference between the models is not just due to noise.

DISTANCES IN QUESTION VECTORS

The SVD model assigns a vector to each question. One way that we can evaluate these vectors is to explore the similarities between them. To that end, we compute the average cosine similarity between the vectors of questions with the same content tag shown in Table 4. These tags describe the content of the question. Each question can have more than one tag, and not all questions have tags. The average similarity over all question vectors was 0.086. Table 4 shows that the average cosine similarity between problems with similar tags is significantly higher, suggesting that the SVD model embeds similar problems closer together.

Tag:	Class	function	return	loops	bool	for	if	str
MC	0.28	0.20	0.29	-	0.20	0.40	-	-
Code	0.27	0.21	0.26	0.17	0.24	0.34	0.23	0.21

Table 4. Average Cosine Similarity of Question Vectors (overall avg=0.086)

CONCLUSION AND FUTURE WORK

We explore a collaborative filtering system for recommending revision questions to CS1 students, and compared the offline performance of the SVD and SVD++ approaches. We also show that the embedding space of the questions is meaningful. Our ultimate goal is to test the efficacy of this collaborative filtering based recommendation system in improving CS1 student performance. We intend to compare the collaborative filtering recommender to a random recommender, and compare real student performance across various term tests.

Acknowledgements

We would like to thank Andrew Petersen for providing access to the PCRS data, and Brian Li, Scarlett Tran and Hassaan Mustafa for development and logistical support.

REFERENCES

[1] Tyne Crow, Andrew Luxton-Reilly, and Burkhard Wuensche. 2018. Intelligent tutoring systems for programming education: a systematic review. In *Proceedings of the 20th Australasian Computing Education Conference*. 53–62.

[2] Nicolas Hug. 2017. Surprise, a Python library for recommender systems. http://surpriselib.com. (2017).

[3] Andrew Petersen. 2013. PCRS. https://mcs.utm.utoronto.ca/ pcrs/pcrs/. (2013). Accessed: 2019-01-20.

[4] Guy Shani and Bracha Shapira. 2014. Edurank: A collaborative filtering approach to personalization in e-learning. *Educational data mining (2014)* (2014), 68–75.

[5] Leen-Kiat Soh, Todd Blank, LD Miller, and Suzette Person. 2005. ILMDA: An intelligent learning materials delivery agent and simulation. In *2005 IEEE International Conference on Electro Information Technology*. IEEE, 6–pp.