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.

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.
will find challenging, we treat the number of attempts
Since we wish to recommend questions that a student
higher the attempt, the more challenging the question.
for the student perceived difficulty of the question: the
question until they obtained full marks. This value is a proxy
number of times that each student attempts each ques-
tion until solving the question, with the maximum allowed at-
tempt 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 dif-
ficulty 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 differ-
ent 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 rat-
ings, 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.

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

Table 3 shows that SVD model performs marginally bet-
ter than other models when predicting user ratings. The
baseline is strong, since the variation in question diffi-
culty 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 ques-
tions 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 simi-
larity 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 recom-
mending revision questions to CS1 students, and com-
pared 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 rec-
ommendation system in improving CS1 student perform-
ance. 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


