Solving Mathematical Exercises: Prediction of Students' Success

Sebastian Wankerl[1,2], Gerhard Götz[2], and Andreas Hotho[1]

[1] DHBW Mosbach, Lohrtalweg 10, 74821 Mosbach
[2] Chair for Computer Science X, Am Hubland, 97074 Würzburg
sebastian.wankerl@uni-wuerzburg.de
gerhard.goetz@mosbach.dhbw.de
andreas.hotho@informatik.uni-wuerzburg.de

Abstract. In educational settings, recommender systems can help to choose the right exercises a student should be given for training. To make good decisions, the system should be able to estimate how successful a student would answer a recommended exercise. In this work, we study the performance of convolutional neural networks and collaborative filtering for estimating students' success. We show that we can distinguish between correctly and wrong processed exercises with a precision of up to 64% while training on a small corpus of 712 user interactions.

Keywords: Recommender systems, Technology enhanced learning

1 Introduction

Recommender systems are widely used for already a long period. While the most prominent areas of application are still e-commerce and entertainment industry[4], several attempts are made for using recommendation algorithms in the area of education[1,2], due to the increasing number of online learning material and massive open online courses (MOOC).

We want to build a recommender system that helps freshmen at university to overcome their weaknesses in mathematics. Since no public data is available for our aim, we built a rule-based system which exploits didactical ontologies[3] to find suitable exercises for a student. Furthermore, this preliminary system makes it possible to collect data we can later use for building a system that better adapts to the individual student.

In this work, we want to present a first step in this direction, namely a machine-learning based approach to estimate whether a student will solve an exercise given by our rule-based system correctly. In particular, we want to detect for all users those exercises that they were given but unable to master. Such failing allows the students to learn more and to broaden their skills instead of simply repeating known content.

2 Educational System and Dataset

Our dataset contains 128 mathematical exercises. With our rule-based system, we collected data from 39 students who contributed a total of 787 processed exercises. However, for our following analysis we only used the data of participants who contributed at least 10 exercises to have enough data for each user. This leaves us with 24 users and 712 processed exercises.

The data points, also called interactions, are represented by 4-tuples (u, i, r_{ui}, t) consisting of a user u, an exercise i and a binary rating label r_{ui} which indicates whether the user u solved exercise i correctly or not. In addition, we keep a consecutive timestamp t which preserves the chronological order of the interactions. The number of contributed interactions by user u is denoted as T_u.

2.1 Training and Test Dataset

In this section we describe how we build the test and training dataset of each student we use for our experiments.

To maximize the amount of data usable for training, we keep all data of the user except the last 5 interactions which we want to predict the user's performance on. Hence, these last five interactions are our test set and denoted as te_u.

We denote the basis training set for user u as tr_u. For example, if we want to predict the performance of user u_1, we keep all interactions from users $u_{2,\ldots,24}$ together with the first $T_1 - 5$ interactions by user u_1 as basis training set.

Since we expect the CNN to require more training data than provided by the basis training set, we decided to create a second training set $\tilde{tr}_u$ by augmenting the basis training set. More precisely, it consists of replicas of the users. They are obtained by creating approximately 20,000 windows w_j of size 5 consisting of random interactions $a_{j_1}, \ldots, a_{j_5} \in tr_u$ such that the following conditions are fulfilled:

1. all interactions $a \in w_j$ come from the same user u
2. the interactions $a \in w_j$ are ordered ascending according to their timestamp t

3 Experimental Setting

To reach our goal of predicting the students' performance on given exercises, we experimented with a Convolutional Neural Network (CNN) architecture as well as collaborative filtering (CF).

3.1 CNN

As stated above, we want to predict the performance of each user u on the 5 exercises contained in te_u. Hence, our CNN architecture maps sequences of 5

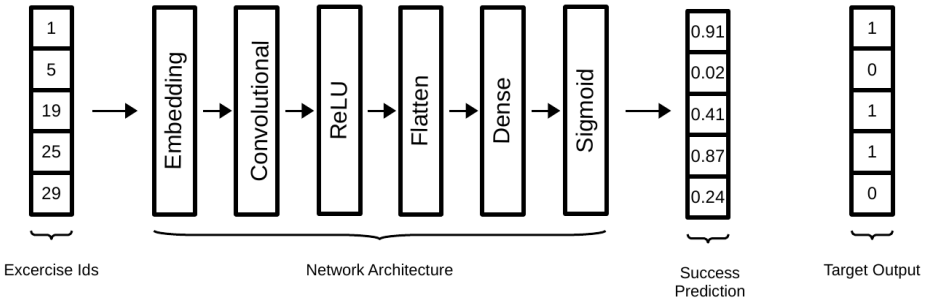

Fig. 1. Visualization of the neural network architecture with example data

exercise ids $(i_1 \dots i_5)$ to their binary performance labels $(r_{i_1} \dots r_{i_5})$. We train the network using the augmented training sets $\tilde{tr}_u$.

Since the inputs to the network are discrete, i.e. high dimensional and sparse ids, they are mapped to a lower dimensional dense vector first using an embedding layer. The dimension of the embeddings is set to 20. We use one convolutional layer, with the number of filters, and kernel size set to 10 and 3, respectively. This configuration worked best out of the explored configurations. We tested the number of filters between 5 and 30 and the number of convolutional layers of 1 and 2. Moreover, we varied the kernel size between 2 and 5.

We flatten the output and put it into a feed-forward layer that applies the sigmoid activation to each output. The overall architecture is depicted in figure 1 alongside a generic input and output sample.

For training, we set a batch size of 32 and apply the ADAM optimizer. Moreover, we use the binary cross-entropy as the loss function since it transforms each element of the sigmoid output layer into an independent probability. In our setting, these are interpreted as the probabilities of the exercises being solvable.

3.2 Collaborative Filtering

We also apply collaborative filtering (CF) using the kNN algorithm with Pearson correlation as similarity measure as this is a long established approach in the field of recommender systems. We chose to consider the $k = 2$ nearest neighbors only as this yields the best results on our dataset. To predict the performance of user u, we fit the algorithm on the original user interactions contained in tr_u. For testing we let the algorithm predict te_u of each user u, equal to the evaluation of our CNN approach.

4 Results

As described in section 3, we evaluated a CNN architecture as well as collaborative filtering on our recommendation setting. We use majority vote of each

Table 1. Results of the Classifiers

Classifier	Precision	Recall	F1	RMSE
CNN	0.64	0.52	0.54	0.48
CF	0.64	0.61	0.59	0.47
Majority	0.56	0.46	0.47	0.53

exercise as a baseline. It assumes that the user u will solve exercise i incorrect if the majority of other users did so.

The classifier's predictions are rounded half away from zero to obtain dichotomous variables of predicted success. The metrics used for evaluation are precision p, recall r, and f_1-score. In addition, we evaluated the RMSE score between the not rounded predictions and r_{ui}. The results are shown in table 1, averaged over all users.

Both classifiers outperform the baseline with regard to all metrics as can be seen in table 1, leading to the conclusion that both classifiers are able to recognize contextual effects in the students' handling of the exercises. Regardless of parameter k, the CF could not make predictions in 13 of 120 exercises we want to predict since no fitting neighbors were available.

With regard to the precision, both approaches work comparably. However, the collaborative filtering yields a considerably higher recall score than the CNN. Hence, the CF approach detects more of the positive items. Overall, the f_1-score of the CF is consequently higher than the one of the CNN.

With regard to the RMSE score, the baseline performs worse than both machine learning based approaches again. Also, it gives further evidence that the predictions of the CF are slightly closer to the true values than the one made by the CF approach.

5 Summary

In this work we trained a CNN and CF to predict a student's success in solving mathematical exercises presented to the student by our tutoring system. The classifiers were given all interactions of the student except the last 5, along with the interactions of all other students. We showed that both classifiers are able to predict the students success more accurately than a majority baseline.

The results suggest that a classifier can help in selecting appropriate exercises for a student. As a further step, we can incorporate it into our rule-based system to help making decisions which exercise to present to a student.

Nevertheless, one has to keep in mind that the results presented here are drawn from a very limited set of students. This could also be a possible explanation why the neural network approach does not yield promising results. It is therefore worthwhile to conduct the described experiments again as soon as more student participated in the training.

Moreover, it can be investigated whether it is possible to boost the quality of predictions if the system could exploit more information than just the correctness

of an exercise, like the student's average time spent with solving the exercises or his overall success.

References

1. Drachsler, H., Verbert, K., Santos, O.C., Manouselis, N.: Panorama of recommender systems to support learning. In: Recommender systems handbook, pp. 421–451. Springer (2015)
2. Henning, P.A., Forstner, A., Heberle, F., Swertz, C., Schmölz, A., Barberi, A., Verdu, E., Regueras, L.M., Verdu, M.J., Pablo de Castro, J., et al.: Learning pathway recommendation based on a pedagogical ontology and its implementation in moodle (2014)
3. Pinkernell, G., Düsi, C., Vogel, M.: Aspects of proficiency in elementary algebra. In: 10th Congress of European Research in Mathematics Education (2017)
4. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys (CSUR) **52**(1), 5 (2019)