Answer Forecasting: ML-Driven Predictions in Lernnavi

Luca Zunino, Elena Grazia Gado, Tommaso Martorella
Machine Learning for Behavioral Data (CS421) - Milestone 4 - EPFL, Switzerland

I. Introduction

In recent years, educational applications have become increasingly prevalent as a means of supplementing traditional classroom learning. Machine learning algorithms have the potential to revolutionize this education sector by providing insights into student learning behavior and enabling the creation of tailored learning experiences. In this regard, developing a machine learning model capable of predicting a user's response could enable the provision of adaptive learning experiences that cater to the needs and abilities of individual students. This has indeed been the interest of various studies, such as [1]. This project aims to explore the development of such a machine learning model for Lernnavi, an educational platform created to support middle and high school students studying Math and German. This report outlines the actions taken thus far and the initial findings in the development of our machine learning model. Additionally, section V is dedicated to detailing the upcoming strategies intended to refine the model towards achieving the final version.
The main research questions we aim at answering in out project are the following:

1) Does the available data allow to extract enough relevant features to predict the user's knowledge and learning?
2) Can we represent the user's knowledge in a compact format (embedding) and use it to predict their behaviour?
3) Can we leverage a language model (GermanBERT in particular) to predict the answers of Lernnavi users?
4) Are the user embeddings helpful to improve the predictions of the model?

II. Feature Extraction and Preprocessing

In order to achieve our ambitious objective of developing a machine learning model capable of predicting a user's response when using the Lernnavi application, we needed to undertake various preprocessing and feature extraction steps. To obtain a comprehensive overview of the available data that could be utilized for our project, we first merged the provided datasets and conducted data cleaning procedures wherever applicable. Afterwards, feature engineering, a crucial stage in our project, was performed. It involved manipulating the datasets containing information from Lernnavi users to extract relevant features that could signify the student's level of understanding and aid in predicting the user's response. The features we decided to compute are the following:

- the mastery level of the user for each topic;
- the mean mastery level of the user;
- the the sum of the scores of the student on the questions answered, normalized by the number of questions answered;
- the sum of the scores of the student on the questions answered weighted on the difficulty, normalized by the number of questions answered;
- the mastery level of the user for each topic.

It must be noted that the difficulty of each question and the associated scores were also extracted from the Lernnavi datasets. The combination of these factors will provide a comprehensive understanding of the student's knowledge and aptitude for learning. Furthermore, the final feature provides insights into the student's level of proficiency in the different topics. Limitations of the computed features will be discussed in section V, together with the steps that will be taken to overcome them.

III. Student Embedding

The second step of the pipeline consists of the creation of compact student representations called embeddings. With this step, we aim to capture the features of the student in a compact and computationally manageable shape that will hopefully be a meaningful representation of the student's current knowledge state. For this purpose we use autoencoders to generate the embeddings. Autoencoders are a type of (usually) symmetric neural network that learns to compress and reconstruct the data with a minimal loss thanks to their bottlenecked double funnel architecture.
While the mean mastery level and the scores are features easily usable to train the network, the topics masteries require an intermediate step. To incorporate the topic information into the input features, we first represent the topic with a high-dimensional embedding using a sentence transformer based on *paraphrase-multilingual-MiniLM-L12-v2*, such that similar topics have similar vectors. Each topic embedding has a size of 384. To combine the 31 topic embeddings with the mastery level of the student for each topic, we use a simple method: we first normalize the topic embedding vector to have a unit norm ($||embedding||_2 = 1$), and then we scale it by the student's mastery level for that topic. This way, we obtain a scaled topic embedding vector that reflects both the semantic meaning of the topic and the student's proficiency in it. This way, the final input features for the autoencoder have a size of 11907, which is composed of 3 base features (mean mastery level, score, weighted score) and 31 scaled topic embeddings (384 x 31).
The autoencoder architecture we are using is a simple symmetric multilayer perceptron (MLP) with two hidden layers in both the encoder and the decoder. The size of the hidden layers is 5000 and 1000 for the encoder, and 1000 and 5000 for the decoder. The input layer has a size of 11907, and the output layer has a size of 11907 as well since the autoencoder tries to reconstruct the input data. The bottleneck layer, which is also called the latent layer or the embedding layer, has a size of 512. This is where we obtain the final embeddings for each student.
The autoencoder is trained using mean squared error (MSE) as the loss function, and Adam as the optimizer. The details of the autoencoder training can be found in the notebook "autoencoder.ipynb". We reach a validation loss of $4.7e^{-6}$, however, a more in-depth analysis shows that our embeddings are far from perfect since we obtain a significant reconstruction difference when considering the norm2 of the input and output vectors for training, validation, and test data (average $||input||_2$: 1.67, average $||input - output||_2$: 0.15).

IV. Answer Prediction

Our project aims to determine the most probable answers of Lernnavi users to provide targeted guidance before they attempt to answer questions. To predict a student's most probable answer, our model will have access to the question, possible answers (if available), and relevant student information in the form of embeddings. This poses several challenges, including the need for natural language understanding, processing student embeddings, and adapting the classifier's prediction based on individual students. To address these challenges, we leveraged a pre-trained model, fine-tuning it for answer prediction. After reviewing the literature, we selected the GermanBERT model, part of the BERT family, as the starting point. This model ("dbmdz/bert-base-german-uncased") has been pretrained on an extensive text corpus, learning effective latent representations of sentences in context. This extensive pretraining allows for fine-tuning the model on specific tasks with less computational effort. For the moment, we focused on multiple-choice questions, simplifying performance evaluation due to closed answers. To simplify as much as possible the task, we decided to cast the problem as binary classification: given a question, and N possible answers, the input to the model is in the format [QUESTION] + [i-th ANSWER] for each of the N answers. The corresponding desired output is 0 if the student did not select the i-th answer or 1 if the student selected the i-th answer. Another critical aspect that needs to be decided is how to combine the embeddings and the model's representation of the question+answer. The model needs to exploit this rich, high-dimensional additional information about the student to produce an accurate prediction, but at the same time, we do not want to excessively disrupt the representations that the model has learnt during the pretraining phase. In the exploratory analysis, we considered four main approaches (explained in detail in the associated notebook):

- Concatenating student embeddings with the question+answer representation just before the classifier layer (custom model MCQBert1);
- Summing student embeddings with the question+answer representation just before the classifier layer (custom model MCQBert2);
- Summing student embeddings with the question+answer embeddings before the first layer of the model (custom model MCQBert3);
- Concatenating student embeddings with the question+answer embeddings before the first layer of the model (custom model MCQBert4).

In addition to these four approaches, we considered a baseline model without student embeddings, which performs classification based on the question and possible answer.

A. *Results and analysis*

In this part of the analysis, we aim to answer two primary research questions:

1) Can we leverage a language model to predict the answers of Lernnavi users, in order to provide targeted guidance to them while they are still trying to answer a given question?
2) Are the embeddings of the users helpful to improve the model predictions?

To answer the first question, we verified the performance obtained by the baseline model (custom BERT model without student embeddings) after the fine-tuning on the binary classification task described above. To perform fine-tuning, splitting the dataset into a training set and a validation set has been necessary. Since a single MCQ has been answered several times by different students, many rows corresponding to the same MCQ are present in the dataset. To answer the research questions, we first decided to split the dataset line by line without enforcing that a single MCQ can be only in the training or validation datasets. In other words, the model has been evaluated on its ability to predict the student's answers to MCQs that saw before but not for that particular student. We should, in any case, be attentive to data leakage, and we, therefore, decided also to verify whether the model can generalise well to MCQs never seen before (see last result). The dataset processing, the training/validation pipelines, the model definitions, and the complete results obtained by the model for each training epoch can be found in the Jupyter Notebook "GermanBERT.ipynb", and we will therefore not repeat everything here for the lack of space. The metrics which we considered in order to assess the performance of the model, and to verify whether a language model can be used to predict the answers of Lernnavi users, are the following:

- Accuracy score, which measures the proportion of correct predictions out of the total number of predictions.
- F1 score, which is the harmonic mean of precision and recall, and allows balancing both false positives and false negatives.
- Matthews Correlation Coefficient (MCC), which is a measure of the quality of binary classification. This metric is effective even if the classes are strongly unbalanced, and it is a correlation coefficient value between -1 and +1 (+1: perfect prediction, 0: average random prediction, -1: inverse prediction).

We consider the best results obtained by the baseline algorithm on the fine-tuning task: the accuracy score was 0.820, the average F1 score was 0.8035 (F1 score for class 0: 0.860, F1 score for class 1: 0.747), and the MCC was 0.607. These results clearly indicate that even a baseline model that does not consider information about the user with which the question is associated can reliably predict the answers to MCQs. Therefore, it seems possible to leverage a language model to predict the answers of Lernnavi users, and using a BERT model seems a good starting point. To verify whether the embeddings of the students can help the classifier predict the answers and improve the performance of the model, we then considered the four models detailed above, MCQBert1-4. These four models all consider student embeddings, although in different ways. We repeated the same fine-tuning procedure detailed for the baseline model, and we considered the models' performances by analysing the same metrics (accuracy score, F1 score, MCC). Once again, the complete results are available in the notebook, while here, we indicate the best performance obtained by the model:

- MCQBert1 $\rightarrow$ Accuracy: 0.817, avg. F1: 0.8045, MCC: 0.610.
- MCQBert2 $\rightarrow$ Accuracy: 0.817, avg. F1: 0.8040, MCC: 0.609.
- MCQBert3 $\rightarrow$ Accuracy: 0.821, avg. F1: 0.8070, MCC: 0.614.
- MCQBert4 $\rightarrow$ Accuracy: 0.821, avg. F1: 0.6140, MCC: 0.607.

From these results, it is possible to notice that, apart from MCQBert4, all the other custom models which considered the embeddings outperformed the baseline model. However, the improvement is quite limited (the best model among the four considered, MCQBert4, got a percentage improvement of +0.43% over the baseline average F1 score and +1.15% over the baseline MCC). The model which performed the best on the validation set was, perhaps surprisingly, MCQBert3, in which the student embeddings are summed at the model's input. Our initial expectation was that this model would have been the worst among all four, as summing the embeddings with the embeddings of the question+answer at the model's input directly disrupts the input which the model is expecting after the pretraining phase. The possible reasons behind this result are explained in the notebook. Therefore, by analysing these results, it is possible to conclude that the embeddings can indeed be beneficial to the model. However, the minimal improvements over the baseline indicate that the model is still not taking significant advantage by the embeddings, and that a significant amount of work remains to be done on the student embeddings and the model optimisation. Finally, we verified whether the model can generalise to never-seen-before MCQs. This approach is much more challenging for the model, as it lacks any prior history related to the specific questions it is attempting to evaluate, and it cannot rely on the answers provided by students with similar profiles to the one for whom it is predicting the answer. Coherently with the task being significantly more complex, the model's performance drastically dropped. Indeed, for the best epoch, the accuracy score has been 0.561, the average F1 score 0.528 (F1 score for class 0: 0.652, F1 score for class 1: 0.404), and the MCC has been 0.057. The model's performance under this more challenging situation is really bad, just slightly better than a random guesser. This is not necessarily an issue by itself, as in a real-life scenario, the model would have a history for most of the MCQs present in Lernnavi, and it could therefore exploit this history to learn associations between the predictions of given users and the profiles/embeddings of that users. Combined with the fact that the models considering embeddings only slightly outperform the baseline one, the poor performance in this configuration is a hint that the model can extract a minimal amount of information from the student embeddings while it relies primarily on previous history for predictions.

V. Future Directions

A. *Feature extraction and preprocessing*

As previously stated, there are certain limitations associated with the features that have been extracted. Specifically, it should be noted that the mastery scores extracted are the most recent ones in the available student progress timeline. At the same time, the scores for each student were computed with all the available data. We are aware that there is a certain (little) amount of "data leaking". However, it is our aim to fix those issues in the final version of our project and to determine an effective train-test split to accurately validate how the model generalizes on new data. More specifically, we would like to test the following generalizations (in increasing order of difficulty):

- the pair (known student, question not seen for that student);
- the pair (unseen student, known question);
- the pair (unseen student, unseen question).

B. *Student embedding*

As of now, our approach is severely limited by two main factors:

- The lack of use of the student's answered questions;
- The lack of temporality.

The former prevents the model from precisely learning how the student actually performs on real examples and thus, having a more in-depth understanding of the student's knowledge and how it's distributed across topics and subtopics. The latter, instead, prevents the model to understand how the student's knowledge is evolving and thus capturing the dynamics of learning. We plan to incorporate these two missing factors by using knowledge tracing techniques based on LSTM to analyze the student answer patterns.

C. *Answer prediction*

The analysis suggests that student embeddings may enhance classifier performance, as models using embeddings slightly outperformed the baseline. However, improvements were not as significant as anticipated, indicating further work is needed to refine the custom GermanBERT model. This will be our priority in the second part of the project. Additionally, we aim to extend our model to open questions, though evaluating semantic similarity between predicted and actual answers presents challenges. We will likely use model-based metrics, such as BERTScore, to measure similarity in candidate and reference sentences. The detailed analysis of future directions for answer prediction is available in section 11 of the notebook "GermanBERT.ipynb".

References

[1] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/bac9162b47c56fc8a4d2a519803d51b3-Paper.pdf

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

VI. Contributions and teamwork reflection

A. *Contributions*

Milestone 4 was a collaborative effort by our team, and each member made significant contributions to the project. The team members' specific contributions were as follows:

- Elena Grazia Gado: Elena was responsible for feature extraction and preprocessing. She extracted relevant features from the raw data and preprocessed them to be used in the GermanBERT model.
- Tommaso Martorella: Tommaso worked on the student embedding component, which aimed to represent students in a way that captured their abilities and knowledge.
- Luca Zunino: Luca was responsible for the answer prediction part of the project. He developed and trained the GermanBERT to predict a user's response based on the input features and student embeddings.

These three contributions have been equal in terms of workload and contribution to the project. Furthermore, all team members cooperated in writing the report and defining the research questions.

B. *Reflection on teamwork*

The teamwork for Milestone 4 has been quite successful, with no significant issues arising. A clear division of tasks allowed each team member to focus on their specific area of expertise, resulting in efficient progress. Regular communication and collaboration facilitated a smooth integration of each individual's work into the final report. Additionally, the team's ability to work together to define research questions and share ideas contributed to a cohesive project. Overall, the team has displayed strong teamwork and synergy thus far.