Towards Student Behaviour Simulation: A Decision Transformer based Approach

Zhaoxing Li$^{1[0000-0003-3560-3461]}$, Lei Shi$^{2[0000-0001-7119-3207]}$, Yunzhan Zhou$^{1[0000-0003-1676-0015]}$, and Jindi Wang$^{1[0000-0002-0901-8587]}$

^{1} Department of Computer Science, Durham University, Durham, UK
^{2} Open Lab, School of Computing, Newcastle University, Newcastle upon Tyne, UK
`{zhaoxing.li2, yunzhan.zhou, jindi.wang}@durham.ac.uk`
`lei.shi@newcastle.ac.uk`

Abstract. With the rapid development of Artificial Intelligence (AI), an increasing number of Machine Learning (ML) technologies have been widely applied in many aspects of life. In the field of education, Intelligence Tutoring Systems (ITS) have also made significant advancements using these technologies. Developing different teaching strategies automatically, according to mined student characteristics and learning styles, could significantly enhance students' learning efficiency and performance. This requires the ITS to recommend different learning strategies and trajectories for different individual students. However, one of the greatest challenges is the scarcity of data sets providing interactions between students and ITS, for training such ITS. One promising solution to this challenge is to train "sim students", which imitate real students' behaviour while using the ITS. The simulated interactions between these *sim students* and the ITS can then be generated and used to train the ITS to provide personalised learning strategies and trajectories to *real students*. In this paper, we thus propose SimStu, built upon a Decision Transformer, to generate learning behavioural data to improve the performance of the trained ITS models. The experimental results suggest that our SimStu could model real students well in terms of action frequency distribution. Moreover, we evaluate SimStu in an emerging ITS technology, Knowledge Tracing. The results indicate that SimStu could improve the efficiency of ITS training.

Keywords: Student Modelling· Decision Transformer · Intelligent Tutoring Systems · Behavioural Patterns

1 Introduction

The recent COVID-19 has significantly impacted people's educational activities, which promoted the Intelligent Tutoring System (ITS) to achieve outstanding development. Data-intensive approaches have been proposed for ITS to improve the quality of education services [23]. However, these need to be powered by data-hungry machine learning models, whose performance relies heavily on the size of training data available [22]. Moreover, similar to the scarcity of labelled data in many Artificial Intelligence (AI) fields, the shortage of student behavioural data has become one of the greatest challenges for ITS advancements [24]. Our work thus aims to tackle this challenge by answering the following research question:

How to create adequate high-fidelity and diverse simulated student behavioural data for training ITS?

In this paper, we propose a Transformer-based approach based on our previous work [11]. The intuition of SimStu (shown in Fig. 1) is that after an ITS collects *a small amount* of real student behavioural data in the early stage development, it feeds the data into a generator, which produces *a large amount* of simulated student behavioural data. These simulated data can then be combined with the real student behavioural data, to train the ITS, thus improving ITS training. The generator, which we call "SimStu", is built upon the Decision Transformer [3]. In the subsequent research, to train and evaluate our SimStu model, we used the EdNet dataset[3], which is the largest student-ITS interaction benchmark dataset so far. Moreover, we improve the model's performance by modifying the input and hyperparameters.

In this work, we proposed an upgrade version of the SimStu, which obtains better performance. The results suggest that our method could simulate real students well on the metrics of action distribution. In addition, we applied our method in real educational scenarios, Knowledge Tracing models. Knowledge Tracing (KT) is a method that predicts the student's next action based on their previous ones. Many ITS use KT models' prediction results to improve the student learning experience, e.g., giving recommendations for the next learning materials. Therefore, we applied our method's generated data in the state-of-the-art KT models, i.e., SAINT, SSAKT and LTMTL, to evaluate the performance of our model. The experimental results show that our method could improve the KT model's performance.

The main contributions of this paper lie in the following three aspects:

1. We propose a student learning behaviour simulation approach (SimStu) based on the Decision Transformer, aiming to provide adequate training data for ITS.
2. Our experiments demonstrate that a trained SimStu model can simulate real student behaviour well and outperform imitation learning based models.
3. We evaluate SimStu in a real ITS education scenario - applying SimStu in three state-of-the-art KT models (SAINT, SSAKT, LTMTL), and the results show that our approach could improve the performance of each KT model.

2 Related Work

2.1 Student Modelling

With increased attention to personalised learning, the traditional one-size-fits-all method can no longer satisfy user needs [2]. In offline scenarios, personalised

[3] http://ednet-leaderboard.s3-website-ap-northeast-1.amazonaws.com

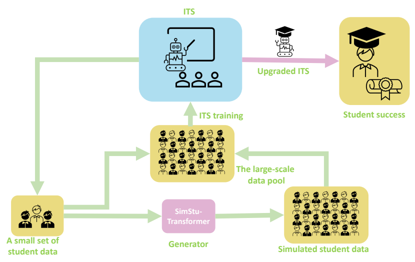

Fig. 1. The intuition for the proposed SimStu pipeline in ITS.

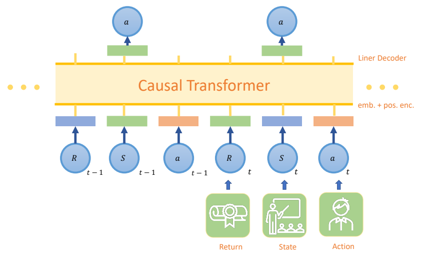

Fig. 2. SimStu architecture.

learning can be supported by teachers in various ways. For example, a teacher can gain valuable information about their students, by observing their learning process and interactions and then design the most suitable and beneficial learning strategy for them [7]. However, the lack of teacher-student interactions in online learning environments makes the personalisation process extremely difficult [1, 12]. In such online scenarios, student modelling can and has been applied, as a powerful tool to combat this issue [5]. Thereby, in the current study, we take advantage of the benefits of group-level student modelling and train our system using the learning data from a large number of individual students to learn the patterns of student learning in the system. This can then enable the system to recognise "optimal" learning behavioural patterns, which lead to the better student experience, performance, and learning results, as well as "poor" learning behavioural patterns, which may result in failure, thus recommending not only personalised but also optimal learning trajectories to the students, or providing a reminder of progressing to potential failure. To achieve this objective, it is crucial to have a decent quality and quantity of training data to feed to 'data-hungry' machine learning models.

2.2 Knowledge Tracing

Knowledge Tracing (KT) is a common method of personalising learning strategies for individual students. It predicts whether a student has the capability to master a new piece of knowledge, by tracing the student's current knowledge state, which depends on past learning behaviour. The two major KT approaches are Bayesian Knowledge Tracing (BKT) and Deep Knowledge Tracing (DKT) [15].

BKT is a probabilistic method for student model generalisation [9]. It uses the Hidden Markov Model (HMM), to model their knowledge state as a set of binary parameters, each of which indicates whether a single Knowledge Concept (KC) has been understood or not [6]. DKT considers knowledge tracing as a sequence prediction problem. It uses Recurrent Neural Network (RNN) to model a student's knowledge state in one summarised hidden vector [15]. DKT is powerful for capturing a complicated depiction of human learning. However, the parameters of the DKT model are non-interpretable [10], which may result in students distrusting the system and teachers being unable to understand student behaviour. Additionally, when dealing with sparse data, DKT may encounter the problem of not generalising well [8]. The main limitation of BKT and DKT is that they both rely on a huge amount of students' historical learning data [13]. Different from BKT and DKT, our approach generates *simulated* student learning data, thus not relying on a huge amount of *historical* data, and more importantly, the simulated data can be visualised in statistical charts, showing student's learning behavioural patterns and thus being able to mitigate the KT model's limitation of non-interpretability.

2.3 Transformers

Transformers have risen to prominence in the field of deep learning in recent years, particularly in natural language processing and image generation tasks [21, 14]. A Transformer is an encoder-decoder Sequence2Sequence architecture to model sequential data, which consists of stacked self-attention layers.

Before self-attention was introduced, the best-in-class architecture was the seq2seq model [18], with an attention component from the decoder to align weights to input positions in the encoder, deciding how much information to retrieve from each position of inputs. Based on the Transformer architecture, Chen *et al.* [3] proposed the Decision Transformer, which abstracts the reinforcement learning problem, as a sequence modelling objective. The key in this algorithm is to generate actions based on *desired returns in the future*, rather than rewards in the past, and they proposed feeding a sequence of returns-to-go (sum of future rewards) $\widehat{R}_t = \sum_{t'=t}^{T} r_{t'}$ into the model. This model first learns a linear layer for each in returns-to-go, state, and action, to project them to the embedding dimension, followed by a layer normalisation. A time-step embedding is also learned and added to the tokens, which are then fed into a GPT [17] architecture, with the goal of generating future actions. Our proposed Sim is built upon this model. We feed the sequence of interactive data between students and the ITS into the Decision Transformer, to generate simulated student behaviour data.

3 Method

3.1 Architecture

The proposed SimStu is built upon the Decision Transformer [3] originally proposed by Chen *et al.* It consists of an encoder and a decoder and models the joint distribution of the sequence of student returns-to-go, states, and actions. Fig. 2 illustrates the architecture. It separates student interactive trajectory sequences into two parts: one is used as the input embedding of the encoder, and the other is used as the output embedding of the decoder [21]. Then, the encoder takes the first part of the trajectory sequence embeddings as input and passes an output trajectory to the decoder. The decoder accepts a shifted embedding trajectory as input to produce the final output trajectory.

3.2 Dataset

The dataset used in our experiment is EdNet [4] - the largest student-ITS interaction benchmark dataset in the field of ITS. It contains more than 780K students' data extracted in South Korea over two years by a multi-platform ITS called SANTA[4]. EdNet consists of four hierarchical datasets, classified according to the number of interactions. We conducted our experiments based on EdNet-KT4, which includes problem-solving logs. Compared to KT-1 to KT-3, KT-4 provides the finest detailed interaction data, allowing access to specific features and tasks.

3.3 Trajectory Representation

The key desiderata of selecting the model features are to provide the algorithm with meaningful information to generate *the most likely trajectories*. We replaced the timestamps with the difference between the individual timestamps, i.e., the time between switching actions. The single timestamp could contain little information, and the time values in the UNIX system that generated them are large. We thus reduced the large UNIX time integers to small values, which also are more suitable for training. Furthermore, we removed from the modelling data types with very sparse data, where it is difficult for the Decision Transformer model to learn anything from the small number of values actually presented in the data. For instance, as *cursor_time* is sparse, with a usual value of *NaN*, we removed *cursor_time* from the data. *action_type* is used to imitate students' behaviour, denoted by a in the Decision Transformer Trajectory τ. *user_answer*, denoted by R, is used for evaluating student performance, thus partitioning them into groups. We examined whether the student's answers (options of a, b, c, and d) matched with the correct answers: if yes, they received a positive reward of 1; and if no, they received a reward of 0. *item_id* is used for evaluating the feasibility of the learning paths, which takes as the state of the student and is denoted by s. Due to the fact that *user_id* does not affect or represent student behaviour, we chose to generate it randomly, after the SimStu generation procedure ended.

3.4 Experiments

The SimStu was implemented using the Pytorch framework and trained on an Nvidia RTX 3090 GPU. We used the Adam optimiser with a batch size of 64. We set Adam betas as (0.9, 0.95). The initial learning rate was 0.0006, and the dropout rate was 0.1. To evaluate the proposed SimStu, we conducted three experiments.

In the first experiment, we compared the simulated data generated by the SimStu model with the original data. More specifically, we examined the average number of actions for the generated and original data amongst the five

[4] https://www.riiid.co/kr

student groups. Furthermore, we compared the similarity of the generated data and the original data using the Pearson product-moment correlation coefficient (PPMCC). PPMCC is a measure of the linear correlation between two variables [16]. A high PPMCC value in the experiment means a high correlation between the original and the generated data and thus indicates our SimStu can simulate student behaviours well.

In the second experiment, we compared our SimStu with Behaviour Cloning, an imitation learning based method proposed by Torabi [20]. We used RELU as the nonlinearity function, with a standard batch size of 64. We set the initial learning rate as 0.0001 and the dropout rate as 0.1. In this experiment, we compared the similarity of students' "elapsed time" between generated and original data using PPMCC. As in the first experiment, a high PPMCC indicates a high simulation performance.

In the third experiment, we evaluated SimStu using three top-performance KT models selected from the Riiid Answer Correctness Prediction Competition on Kaggle[5], which include SAINT, SSAKT, and LTMTI [6]. In the competition, Kaggle provides a dataset containing 2,500 student records to test models. Each student record contains the student's sequence of discrete learning actions. We thus assume that 2,500 student record is sufficient for KT model training. Therefore, We selected five datasets that contained 500, 1,500, 2,000, and 2,500 student records, respectively. We fed these five datasets into the SimStu models, which then generated another five simulated datasets respectively. The generated data size was equal to the original data size. Lastly, we fed these five mixed datasets into the three KT models respectively to compare whether using SimStu could affect the performance of KT models. The metric we used here is AUC (Area Under Curve).

4 Result and Discussions

Fig. 3 shows the results from the first experiment: the average number of actions performed by the real students (on the left) and by the simulated students (on the right), across all those five groups. This suggests that the distributions between the real student data and the simulated student data share some similar statistical characteristics, i.e., in both real and simulated scenarios: 1) the "very good student" group (Group 1) is the largest group, whilst the "very poor student" group (Group 5) is the smallest group; 2) the "good student" group (Group 2) and the "average student" group (Group 3) have similar sizes; and 3) both the "good student" group and the "average student" group are much smaller than the largest "very good student" group (Group 1), and 4) both the "good student" group (Group 2) and the "average student" group (Group 3) are much larger than the smallest "very poor student" group (Group 5). However, the only difference is that in the Real Students scenario (on the left), the "poor student" group (Group 4) is the second smallest group and smaller than both the "good

[5] https://www.kaggle.com/code/datakite/riiid-answer-correctness

[6] http://ednet-leaderboard.s3-website-ap-northeast-1.amazonaws.com

student" group (Group 2) and the "average student" group (Group 3), whilst in the Simulated Students scenario (on the right), the "poor student" group (Group 4) is the second largest group and larger than the "good student" group (Group 2) and the "average student" group (Group 3). Nevertheless, this result suggests that our SimStu model can generate student data similar to real student data.

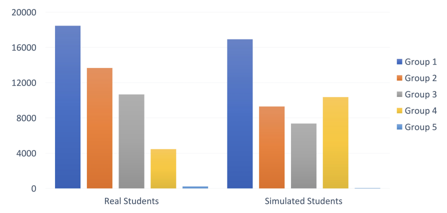

Fig. 3. Action statistics of real student data (left), and simulated student data (right).

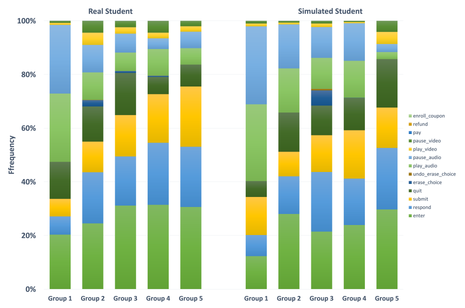

Fig. 4. Action frequency distribution of real student data (left), and simulated student data (right).

As Fig. 3 shows, the SimStu performed better in simulating the behaviour of students with higher grades (i.e. groups 1 ("very good") to 3 ("average")) than for lower grades students (i.e. groups 4 "poor" and 5 ("very poor")). This is in line with the difference in the amount and the frequency of actions. The reason may be that students who study better generally spend a longer time interacting with the ITS, compared to students with relatively poor learning performance. This pattern makes many actions sparse and the causal relationship between actions weak, so the model cannot understand students' behaviours well. To paraphrase Tolstoy's words, "All good students may behave alike, but all poor performance students have their own reasons" [19].

Fig. 4 shows the action frequency distribution of the real student data (on the left) and the simulated student data (on the right). This result shows that the simulated data generated by our SimStu is similar to the real data in major action frequencies. For example, the main actions of the generated data, such as *respond*, *enter*, *play_ audio*, and *submit*, have similar frequencies in each group. However, there are some differences in the actions that occur less frequently, such as *pay* and *undo_ erase_ choice*. The resulting PPMCC value of all actions is equal to 0.714, which suggests that the simulated student data and the real student data are 71.4% similar in the average distribution of actions. The result suggests that simulated data is statistically similar to real data.

In the second experiment, we fed the same training data and test data to the Behaviour Cloning model, which generated 600 students' trajectories data (a total of 4,413,561 actions). The PPMCC value of the SimStu simulated data versus the real data is 0.762, while the PPMCC value of the Behaviour Cloning model simulated data versus the real data is 0.683. This indicates that the SimStu simulated data is more similar to the real data, which suggests that our SimStu model outperforms the Behaviour Cloning model. This result may be due to the fact that when processing sequential student behavioural data, the student actions sequence context allows the SimStu to identify which policy can result in an action that promotes better learning states and improve training dynamics.

In the third experiment, we evaluated SimStu using the three state-of-the-art KT models. Fig. 5 shows the pairwise AUC comparisons of these three KT models trained on the original datasets (SAINT, SSAKT and LTMTL, in blue) and trained on the mixed dataset (SAINT*, SSAKT* and LTMTL*, in orange). In particular, the curves of SSAKT* and LTMTL* are constantly higher than those of SSAKT and LTMTL. The curve of SAINT* is higher than that of SAINT in every dataset, except for the dataset size of 3,000. The results suggest that our method could improve the performance of KT models (AUC, in particular).

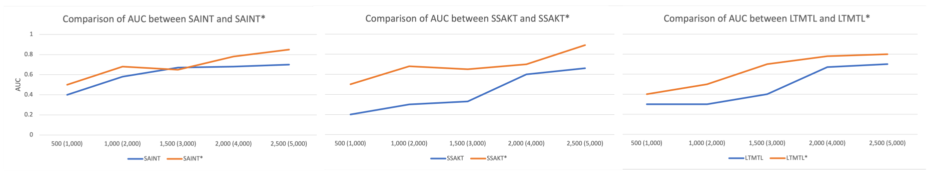

Fig. 5. Pairwise AUC comparisons of the three KT models trained on only original students' data (SAINT, SSAKT, LTMTL, in blue) and trained on the mixed dataset (SAINT*, SSAKT*, LTMTL*, in orange). On the horizontal axis, 500, 1,000,...,2,500 indicate that the grey curve model uses the original dataset, and (1,000),(2,000),...,(5,000) indicate that the red curve model uses the mixed dataset.

5 Conclusion

In this paper, we have proposed SimStu, a Transformer-based approach to simulating student behaviour, aiming to tackle the challenge of the scarcity of datasets for training ITS. We used the EdNet data to train the SimStu model, which generated learning behaviour data that could simulate the learning trajectories of different students. This method could be implemented in an ITS, such that ITS starts with collecting a small amount of student data, then uses our method to generate a large amount of simulated student data, mixes the original data and the generated data to build a new dataset, and finally uses the new dataset to train the ITS and improve its performance. The experimental results showed that SimStu could simulate the students' behaviour data well in terms of *action distribution*. Moreover, we evaluated SimStu by using three state-of-the-art KT models. The results indicated that our method could improve the performance of KT models.

References

1. Bouhnik, D., Marcus, T.: Interaction in distance-learning courses. Journal of the American Society for Information Science and Technology **57**(3), 299–305 (2006)
2. Brusilovsky, P.: Adaptive hypermedia for education and training. Adaptive technologies for training and education **46**, 46–68 (2012)
3. Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., Mordatch, I.: Decision transformer: Reinforcement learning via sequence modeling. arXiv preprint arXiv:2106.01345 (2021)
4. Choi, Y., Lee, Y., Shin, D., Cho, J., Park, S., Lee, S., Baek, J., Bae, C., Kim, B., Heo, J.: Ednet: A large-scale hierarchical dataset in education. In: International Conference on Artificial Intelligence in Education. pp. 69–73. Springer (2020)
5. Chrysafiadi, K., Virvou, M.: Student modeling approaches: A literature review for the last decade. Expert Systems with Applications **40**(11), 4715–4729 (2013)
6. Corbett, A.T., Anderson, J.R.: Knowledge tracing: Modeling the acquisition of procedural knowledge. User modeling and user-adapted interaction **4**(4), 253–278 (1994)
7. Horntvedt, M.E.T., Nordsteien, A., Fermann, T., Severinsson, E.: Strategies for teaching evidence-based practice in nursing education: a thematic literature review. BMC medical education **18**(1), 1–11 (2018)
8. Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: 2018 IEEE International Conference on Data Mining (ICDM). pp. 197–206. IEEE (2018)
9. Kasurinen, J., Nikula, U.: Estimating programming knowledge with bayesian knowledge tracing. ACM SIGCSE Bulletin **41**(3), 313–317 (2009)
10. Khajah, M., Lindsey, R.V., Mozer, M.C.: How deep is knowledge tracing? arXiv preprint arXiv:1604.02416 (2016)
11. Li, Z., Shi, L., Cristea, A., Zhou, Y., Xiao, C., Pan, Z.: Simstu-transformer: A transformer-based approach to simulating student behaviour. In: Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners' and Doctoral Consortium: 23rd International Conference, AIED 2022, Durham, UK, July 27–31, 2022, Proceedings, Part II. pp. 348–351. Springer (2022)
12. Li, Z., Shi, L., Cristea, A.I., Zhou, Y.: A survey of collaborative reinforcement learning: interactive methods and design patterns. In: Designing Interactive Systems Conference 2021. pp. 1579–1590 (2021)
13. Pandey, S., Karypis, G.: A self-attentive model for knowledge tracing. arXiv preprint arXiv:1907.06837 (2019)
14. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.: Image transformer. In: International Conference on Machine Learning. pp. 4055–4064. PMLR (2018)
15. Piech, C., Spencer, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., Sohl-Dickstein, J.: Deep knowledge tracing. arXiv preprint arXiv:1506.05908 (2015)
16. Puth, M.T., Neuhäuser, M., Ruxton, G.D.: Effective use of pearson's product–moment correlation coefficient. Animal behaviour **93**, 183–189 (2014)
17. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
18. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. Advances in neural information processing systems **27** (2014)
19. Tolstoj, L.N., Gerasimov, V.: Anna Karenina. BHB (1969)

20. Torabi, F., Warnell, G., Stone, P.: Behavioral cloning from observation. arXiv preprint arXiv:1805.01954 (2018)
21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
22. Vincent-Lancrin, S., Van der Vlies, R.: Trustworthy artificial intelligence (ai) in education: Promises and challenges (2020)
23. Weitekamp, D., Harpstead, E., Koedinger, K.R.: An interaction design for machine teaching to develop ai tutors. In: Proceedings of the 2020 CHI conference on human factors in computing systems. pp. 1–11 (2020)
24. Yang, S.J.: Guest editorial: Precision education-a new challenge for ai in education. Journal of Educational Technology & Society **24**(1) (2021)