Detecting Need for Help from Content-Independent Sequential Actions with Deep Learning

Raquel Horta-Bartomeu[1,3], Miguel Arevalillo-Herráez[2[0000−0002−0350−2079]] and Olga C. Santos[1[0000-0002-9281-4209]]

[1] Department of Artificial Intelligence, Computer Science School, UNED
rhorta3@alumno.uned.es, ocsantos@dia.uned.es
[2] Departament d'Informàtica, Universitat de València, Avda de la Universidad s/n, Burjassot, 46100, Spain
and
Valencian Graduate School and Research Network of Artificial Intelligence (valgrAI), Valencia, Spain
miguel.arevalillo@uv.es
[3] Codelearn, S.L.

Abstract. Seeking help when needed during learning is a crucial skill, especially in self-guided learning scenarios. However, some students do not ask for assistance within the learning platform even when they need it. We address the challenge of detecting help-seeking behaviour by learning from student-platform interaction using deep learning models. We depict student behaviour as sequences of actions in the course interaction log and evaluate six prediction deep learning models. CNN models trained on recurrence plots showed potential for detecting help-seeking behaviour solely from action sequences, making it more versatile than methods relying on student answers and tasks contents. We foresee potential applications including real-time student support and teacher's alerts.

Keywords: help-seeking, deep learning, CNN, recurrence plots.

1 Introduction

The ability to effectively seek help is crucial in self-guided learning scenarios. While some problematic situations are known both to avoid help and to abuse help as classified in [25], we focus on those that avoid using help. In particular, students may not ask for help even when they need it [1]–[3]. When a teacher is not always present and the student needs some level of self-discipline, not asking for help might be problematic as the student could end up wheel-spinning [4] or abandoning the task. The longitudinal nature of student-platform interaction leads us to think that taking into account the sequential context could be useful for analysing help-seeking behavior using deep learning models. There is literature on help-seeking including sequential data, and some works have focused on performance prediction or have centred on specific knowledge topics. We present an alternative approach to identify undiscovered scenarios where students might require assistance, independent of task content, by examining behavioural patterns in help-seeking actions that consider the sequential nature of student-platform interaction. We aim to answer the research question "Can we detect help-seeking actions from log sequences in an e-learning system?". Fundamentally, this research is encouraged by an ultimate motivation: "Can we identify students who need help but do not ask for it by learning from those who do ask for it?".

In this work, we represent student-platform interactions as sequences of actions and compare six prediction deep learning models to detect help-seeking behaviour. Our approach differs from existing work by joining three main aspects: 1) **Help-seeking behaviour**: we have found works that linked the need for intervention with performance or student knowledge instead of analysing the behaviour surrounding actual help-seeking actions [6]–[8]; 2) **Time component**: we have found works that used cumulative data (e.g. number of attempts) to predict the need for intervention instead of taking into account the temporal context [4]; and 3) **Topic-independence**: we have found works that did take into account the temporal context but focused on the content of student answers for specific tasks [7]–[9].

We aim to predict help-seeking by analysing the sequential nature of student behaviour without relying on student answers or the contents and specifics of the task in hand. We tested our method using data collected over a span of two years from 1263 students working in real, everyday learning scenarios. We foresee potential applications including teacher alerts and real-time student support as well as insights for educators.

2 Related work

Knowing when to ask for help is important [25-27]. It can improve resilience and efficacy [10], [11] contribute to the development of independent and self-regulated skills [3], [12]. Student performance and knowledge have been widely used to determine when to intervene. Levy et al. [3] discovered that prior knowledge could be linked to the predisposition to ask for help and observed that the anonymity behind some computer-based learning environments promoted help-seeking. Mao [7], and Mao and Marwan [8] were able to identify students who needed intervention by predicting whether they would fail to complete their programming task. While the results were positive, this method is very contingent on the specifics of the task at hand (i.e. programming steps) and required task solutions to be available and encoded in a specific format.

Other studies have focused on detecting specific situations known to be problematic. Mu et al. [4] considered a student to be wheel-spinning when they consecutively failed a task over a certain number of times. Their approach was focused on prior attempts, prior knowledge and performance on other activities in the curriculum. Thus, rather than analysing the current student session to detect if a student is showing signs of having trouble, they predicted whether the student would fail to finish the next task. Chou et al. [2] also used a preset definition of needing help when developing a system that negotiates with a student whether they need help or not. They predefined a set of heuristics to assess the need for help.

3 Methodology

3.1 Data acquisition

Data used in this research was provided by Codelearn S.L., a programming and computational thinking academy whose main students are children. The data were collected between July 2021 and January 2023 and were fully anonymised to avoid any personal identification. The dataset consisted of a collection of event sequences representing 10 types of interactions defined in a domain and course-independent way: writing, deleting and pasting text, accessing theory pages and exercise instructions, leaving the platform, idling, failing an exercise, entering an exercise and performing a task (in this case, executing code). These interactions occurred within an online programming platform where students engaged with coding exercises. Students could write, test, and submit code, receiving automated feedback. If an attempt failed, they could revise and try again. Event sequences were captured in a task-independent manner; without requiring knowledge of the specific problem content or student answers. Sessions were of variable length, with inactivity of 10 minutes marking the end of a session.

This research assumes that some students may not seek help, but not asking for help does not always mean they are not struggling. Accordingly, only sequences from students who asked for help at some point were considered. This follows our reasoning that if a student never asked for help, we cannot know whether she might need it or not. On the other hand, we hypothesise that if a student has sometimes asked for help before, she will probably have done so when she needed it (as she knows that she can do it and how to do it).

Help-seeking was integrated into the platform through two tools: students could unlock predefined hints associated with each exercise, or they could send a written help request to their tutor and await a response. Sequences were labeled based on whether the student engaged in help-seeking—i.e., used at least one of these tools.

The collection phase resulted in over 24.4 × 106 raw events. After preprocessing the data, the final dataset consisted of 108829 sequences comprising 8.15 × 106 events from 1263 students between 7 and 18 years old. 5258 of those sequences (4.8%), displayed help-seeking behaviour, with a median of 2.3%

3.2 Experimental protocol

We followed a two-phase approach. First, we tuned the hyperparameters of each model prototype using at least 300 trials, training on 80% of the data and evaluating on the remaining 20%, using the same split across all studies. Trials were assessed using AUROC, and the best configuration per model type was selected. Importantly, this phase involved tuning model structures and settings—not selecting or transferring learned weights. While this approach avoids parameter leakage, we acknowledge that selecting configurations based on the same dataset could introduce some optimistic bias. A more rigorous method would involve nested CV. We then evaluated the selected configurations using 5-fold grouped and stratified cross-validation (CV), ensuring user separation and class balance across folds. Given the class imbalance, we prioritized metrics such as FPR, TPR, and AUROC over accuracy.

3.3 Sequence Representation

Our dataset consisted of sequences of actions, some of which were quantified representing different metrics. Since the goals is to determine whether including the sequential context can provide further insight to detect help-seeking in comparison to merely using aggregated information, each event frequency within a sequence was used as feature to build a baseline model. The other models used the following representations:

1) **Word embeddings**: Categorical sequences have been extensively studied to represent text. The advances made in the natural language processing (NLP) field can then be adapted to other problems. A clear example is word embeddings, which are capable of capturing context and relationships within the tokens they represent. In this work we used learned embeddings with a DistilBERT transformer [14]. This method of data representation did not include the quantification information of our events.

2) **One-hot encoding**: Serving a similar purpose to embeddings, while it does not convey any context or intrinsic relationship between categories, one-hot encoding is also a method to represent categorical data in a way that machine learning algorithms can interpret. This data representation method is commonly used in a binary manner, in which the vector that represents a token has only a single element set to 1, indicating which category it pertains to. Instead of employing binary vectors, we used quantified encodings that stored event quantification values in the corresponding "hot" vector elements.

3) **Deep-RPs**: Recurrence plots (RPs) are a data representation technique with the potential to reveal sequential patterns. They consist of square matrices and thus can be fed to CNNs as if they were images. The shapes they form are characterised by the recurrence patterns of the sequences they represent. When used with categorical data, RPs lack specificity about category associations. To address this, we introduce deep categorical recurrence plots (deep-RPs), a novel variation of RPs. A traditional RP consists of a single matrix in which each point (x, y) reflects a comparison between the elements at positions x and y of the sequence. When used with categorical data, a binary value indicates if the points share the same category. Without further encoding, the category itself is lost. Inspired by the one-hot encoding method and the convolutional neural network (CNN) image processing technique that uses channels to encode colours, we propose adding depth to RPs to encode the category. Given the original $N \times N$ matrix, an extra dimension is added to encode the category. Thus, when elements x and y belong to the same category, the point (x, y, c), where c is the index corresponding to said category, is set to 1 or a chosen distance value. In this work, apart from the deep-RP, we included an extra channel containing a traditional RP. The authors in [15] used binary (i.e. black and white) RPs. Since they worked with sequenc-

es of IDs, binary RPs were a good solution. If we used this method, we would be losing valuable information about which are the recurrent events. This is the reason for using deep categorical RPs. We have seen the depth (channels) of CNNs being used to encode categorical data in works like [16] and [17]. However, we are yet to find a work that makes use of the CNN channels' depth to encode categorical RPs.

3.4 Model prototypes

Six model prototypes were tested: one fully-connected artificial neural network (ANN), two transformer networks, two CNNs and a model that combined multiple CNNs.

1) **Baseline model**: A deep, feed-forward, fully connected neural network was used as a baseline model that cannot reflect the sequential context of the data, allowing us to compare its performance to those solutions that can. It was trained on aggregated features that summarised the sequences.

2) **Transformers**: Considering the resemblance between event sequence classification and text classification tasks we examined the possibility of reusing DistilBERT [14], a state- of-the-art NLP model, by maintaining its pre-trained encoder and re-training its embeddings to fit our data. Values from quantified events were not used since NLP models lack this ability. The second approach used a transformer encoder trained from scratch on quantified one-hot inputs with positional encoding, training only the encoder and classifier.

3) **CNN**: A 1D CNN was used to classify quantified one-hot encodings and a 2D CNN was trained on deep-RPs. To explore whether the 1D and 2D CNNs could complement each other's performance, an extra model that combined the two of them was built (namely, a stacked CNN). The outputs of each submodel's final linear layer were normalized, concatenated, and passed through fully connected layers to learn from the features extracted by each submodel.

4 Results

Once the best configuration was determined during the hyperparameter-tuning phase, models were built and evaluated using a 5-fold CV. Table 1 presents the results. The CNN models performed better than the transformer models, and they all outperformed the baseline considerably. The highest AUROC was achieved by the stacked CNN but was just slightly higher than 1D-CNN, which is a lighter model.

To apply these CNN models, the trade-offs and misclassification costs for each class have to be considered. That is, how many students to interrupt unnecessarily in favour of helping a student in need. Considering a base system without any detection tool, the ability to detect even a few of the students that need help is an improvement compared to detecting none. Therefore, a threshold that compromises a low percentage of non-help-seeking interactions (yields a low FPR) has to be chosen. For instance, considering the 1D-CNN and the stacked CNNs, up to 50% TPR can be achieved if a maximum FPR of 10% is allowed, and up to 27% TPR with an allowed FRP of 5%. This is to be decided by the educational expert.

Table 1. Results of the 5-fold CV for each model, average AUROC and standard deviation.

Model	AUROC	σ
Fully-connected ANN (baseline)	0.633	0.016
DistilBERT	0.756	0.009
Quant. transformer	0.732	0.012
1D-CNN	0.837	0.005
RP-CNN	0.803	0.011
Stacked CNN	0.839	0.006

5 Explainability

When developing a system to suggest the need for intervention, or even to automatically intervene during learning experiences, providing explanations for the system's decisions is crucial. It is important to understand why a model makes a decision and which student behavior traits help flag a situation. We focused on the last linear layer before the RP-CNN's classifier, and considered its output to be a summary of the features that the model learned about the input. We extracted these features for every RP and applied K-means to cluster them. Some of the clusters' class ratios (Table 2) differed a lot from the baseline ratio (i.e. 4.8% positive). A clear visual difference can be seen between the RPs in Fig. 1. Some of the clusters showed distinctive patterns. For instance, a pattern that appeared frequently in cluster 13 involved failing and editing multiple times before asking for help and exiting the platform after failing an attempt.

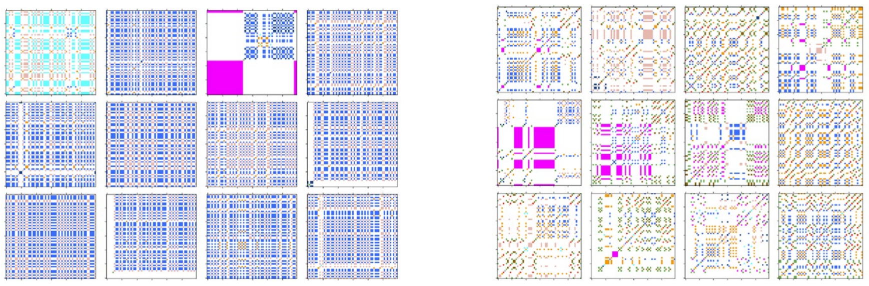

Fig. 1. Samples from clusters 3 (left) and 13 (right)

The cluster analysis revealed that the features extracted by the models successfully differentiated various patterns within the data. The use of recurrence plots proved to be effective, enabling us to visualise these patterns clearly. Some clusters had positive class ratios as high as 33%, nearly seven times the baseline of 4.8%. However, it's worth noting that the majority of samples in these clusters remained negative. While we can reasonably conclude that certain interaction sequences do not exhibit help-seeking behaviour, our confidence is lower when it comes to interactions that do seem to indicate a need for assistance.

Table 2. Number and ratio of samples by class in each cluster.

Clusters	Positive	Negative
13	212 (33.9%)	414 (66.1%)
4	910 (25.0%)	2728 (75.0%)
5	1436 (15.5%)	7815 (84.5%)
1	1801 (9.9%)	16326 (90.1%)
Baseline ratio	4.8% 95,2%	95,2%
0, 7, 10	> 1%, < 4	> 96%, < 99%
2,3,6,8,9,11,12,14	< 1%	> 99%

6 Conclusions and Future Work

Help-seeking has been studied in literature from various angles and knowing when to intervene poses a significant challenge [24-27]. Some approaches are based on how the student's answer to a problem compares to the expected response. Others predefined intervention situations or took a wider scope by considering past attempts and predicting the need for help before the actual attempt. We approach the problem by studying the learning session itself and how the student behaves during this session exploiting the sequential nature of logs interactions and capturing the sequential characteristics of the data with deep learning models. From our experiments, CNN models trained on deep recurrence plots seem to be able to detect students that avoid asking for help but need it. However, further studies need to be done to obtain stronger conclusions.

While we used programming exercises as a case study, the only characteristic specific to programming was the tasks "execute" and no code content was used. The method's core strength lies in identifying behaviour patterns from interactions rather than focusing on tasks' or answers' contents. This could allow applying it to a wide array of educational tasks. The requirement for the method's applicability is a certain level of interaction, such as writing, consulting theory or instructions, and failing.

We consider that this work can contribute to getting a deeper understanding of how students behave when they need help. More implementations could be done to explore potential applications including offering help in real-time, which can be achieved at different levels of disruption. Another application would involve notifying the teacher when a student seems to be struggling through real-time alerts or periodic progress summaries.

Future work could also explore the influence of individual personalities on the decision to seek assistance, with the Big Five model serving as a promising avenue for further investigation. Finally, according to Aleven and Koedinger [18], some students are more interested in getting a task done than learning from it. This is linked to the concept of gaming the system, which has been deeply studied by Ryan Baker [19-22] and is another potential topic to explore using our approach for help-seeking detection.

References

1. I. Roll, R. S. d. Baker, V. Aleven, and K. R. Koedinger, "On the benefits of seeking (and avoiding) help in online problem-solving environments," Journal of the Learning Sciences, vol. 23, no. 4, pp. 537–560, 2014.
2. C.-Y. Chou, K. R. Lai, P.-Y. Chao, S.-F. Tseng, and T.-Y. Liao, "A negotiation-based adaptive learning system for regulating help-seeking behaviors," Computers & Education, vol. 126, pp. 115–128, 2018.
3. R. Levy Cohen and A. Zusho, "Prior achievement in math impacts adolescents' help-seeking behavior in interactive learning environments," New Directions for Teaching and Learning, vol. 2023, no. 174, pp. 65– 71, 2023.
4. T. Mu, A. Jetten, and E. Brunskill, "Towards suggesting actionable interventions for wheel-spinning students." International Educational Data Mining Society, 2020.
5. R. Horta-Bartomeu and O. C. Santos, "A time-aware approach to detect patterns and predict help-seeking behaviour in adaptive educational systems." in 14th International Conference on Educational Data Mining (EDM 2021), 2021, pp. 838–841.
6. A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," User modeling and user-adapted interaction, vol. 4, pp. 253–278, 1994.
7. Y. Mao, "One minute is enough: Early prediction of student success and event-level difficulty during novice programming tasks," in Proceedings of the Twelfth International Conference on Educational Data Mining (EDM 2019), 2019, pp. 119–128.
8. Y. Mao and S. Marwan, "What time is it? student modeling needs to know," in Proceedings of the Thirteenth International Conference on Educational Data Mining (EDM 2020), 2020, pp. 171–182.
9. C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. Guibas, "Learning program embeddings to propagate feedback on student code," in International conference on machine Learning. PMLR, 2015, pp. 1093–1102.
10. S. A. Karabenick and R. S. Newman, Help seeking in academic settings: Goals, groups, and contexts. Routledge, 2013.
11. S. Järvela¨, "How does help seeking help?–new prospects in a variety of contexts," Learning and Instruction, vol. 21, no. 2, pp. 297–299, 2011.
12. Y.-J. Tseng, Y.-H. Lin, G. Yadav, N. Bier, and V. Aleven, "Curio: An on-demand help-seeking system on itextbooks for accelerating research on educational recommendation algorithms," in Proceedings of the Fifth International Workshop on Intelligent Textbooks 2023 co-located with the 24th International Conference on Artificial Intelligence in Education (AIED 2023), 2023, pp. 94–101.
13. R. Agrawal, A. Habeeb, and C. Hsueh, "Learning user intent from action sequences on interactive systems," in Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7 2018, p. 59–64.
14. V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," in Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)., 2019.

15. A. K. Desta, S. Ohira, I. Arai, and K. Fujikawa, “Rec-cnn: In-vehicle networks intrusion detection using convolutional neural networks trained on recurrence plots,” Vehicular Communications, vol. 35, p. 100470, 2022.
16. P. N. Yeboah and H. B. Baz Musah, “NLP technique for malware detection using 1D CNN fusion model,” Security and Communication Networks, vol. 2022, 2022.
17. A. K. Sharma, S. Chaurasia, and D. K. Srivastava, “Sentimental short sentences classification by using cnn deep learning model with fine tuned word2vec,” Procedia Computer Science, vol. 167, pp. 1139–1147, 2020.
18. V. Aleven and K. R. Koedinger, “Investigations into help seeking and learning with a cognitive tutor,” in Papers of the AIED-2001 workshop on help provision and help seeking in interactive learning environments, 2001, pp. 47–58.
19. R. S. d. Baker, A. T. Corbett, K. R. Koedinger, and I. Roll, “Gener- alizing detection of gaming the system across a tutoring curriculum,” in Intelligent Tutoring Systems: 8th International Conference, ITS 2006, Jhongli, Taiwan, June 26-30, 2006. Proceedings 8. Springer, 2006, pp. 402–411.
20. R. S. Baker, “Gaming the system: A retrospective look,” Philippine Computing Journal, vol. 6, no. 2, pp. 9–13, 2011.
21. R. S. d. Baker, A. Corbett, I. Roll, K. Koedinger, V. Aleven, M. Cocea, A. Hershkovitz, A. de Caravalho, A. Mitrovic, and M. Mathews, “Mod- eling and studying gaming the system with educational data mining,” International handbook of metacognition and learning technologies, pp. 97–115, 2013.
22. L. Paquette and R. S. Baker, “Comparing machine learning to knowledge engineering for student behavior modeling: a case study in gaming the system,” Interactive Learning Environments, vol. 27, no. 5-6, pp. 585– 597, 2019.
23. L. Jaccheri, C. Pereira, and S. Fast, “Gender issues in computer science: lessons learnt and reflections for the future,” in 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Comput- ing (SYNASC). IEEE, 2020, pp. 9–16.
24. S. Marwan, A. Dombe, and T. W. Price. 2020. Unproductive Help-seeking in Programming: What it is and How to Address it. In Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20). Association for Computing Machinery, New York, NY, USA, 54–60. https://doi.org/10.1145/3341525.3387394
25. Maniktala, M., Cody, C., Barnes, T. et al. Avoiding Help Avoidance: Using Interface Design Changes to Promote Unsolicited Hint Usage in an Intelligent Tutor. Int J Artif Intell Educ 30, 637–667 (2020). https://doi.org/10.1007/s40593-020-00213-3.
26. N. Alam, M. Maniktala, B. Mostafavi, M. Chi, and T. Barnes. 2023. Does knowing when help is needed improve subgoal hint performance in an intelligent data-driven logic tutor? In Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence (AAAI'23/IAAI'23/EAAI'23), Vol. 37. AAAI Press, Article 1818, 15895–15902.
27. Alam, N., Mostafavi, B., Chi, M., Barnes, T. (2023). Exploring the Effect of Autoencoder Based Feature Learning for a Deep Reinforcement Learning Policy for Providing Proactive Help. In: Wang, N., Rebolledo-Mendez, G., Dimitrova, V., Matsuda, N., Santos, O.C. (eds) Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky. AIED 2023. Communications in Computer and Information Science, vol 1831.