Learning Cognitive Models Using Neural Networks

Devendra Singh Chaplot(✉), Christopher MacLellan, Ruslan Salakhutdinov, and Kenneth Koedinger

Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15217, USA
{chaplot,cmaclell,rsalakhu}@cs.cmu.edu, koedinger@cmu.edu

Abstract. A cognitive model of human learning provides information about skills a learner must acquire to perform accurately in a task domain. Cognitive models of learning are not only of scientific interest, but are also valuable in adaptive online tutoring systems. A more accurate model yields more effective tutoring through better instructional decisions. Prior methods of automated cognitive model discovery have typically focused on well-structured domains, relied on student performance data or involved substantial human knowledge engineering. In this paper, we propose Cognitive Representation Learner (CogRL), a novel framework to learn accurate cognitive models in ill-structured domains with no data and little to no human knowledge engineering. Our contribution is two-fold: firstly, we show that representations learnt using CogRL can be used for accurate automatic cognitive model discovery without using any student performance data in several ill-structured domains: Rumble Blocks, Chinese Character, and Article Selection. This is especially effective and useful in domains where an accurate human-authored cognitive model is unavailable or authoring a cognitive model is difficult. Secondly, for domains where a cognitive model is available, we show that representations learned through CogRL can be used to get accurate estimates of skill difficulty and learning rate parameters without using any student performance data. These estimates are shown to highly correlate with estimates using student performance data on an Article Selection dataset.

1 Introduction

A cognitive model is a computational model of human learning and problem-solving which can be used to predict human behavior and performance on the modeled problems. In this paper, we consider cognitive models in the context of intelligent tutoring systems. A cognitive model that matches student behavior provides useful information about skill difficulties and learning rates. This information often leads to better instructional decisions by aiding in problem design, problem selection and curriculum sequencing, which in turn results in more effective tutoring. A common method for representing a cognitive model is a set of *Knowledge Components* (KC) [1], which represent pieces of knowledge, concepts or skills that are required for solving problems.

C. Penstein Rosé et al. (Eds.): AIED 2018, LNAI 10947, pp. 43–56, 2018.
https://doi.org/10.1007/978-3-319-93843-1_4

Cognitive models are a major bottleneck in intelligent tutor authoring and performance. Traditional ways to construct cognitive models such as structured interviews, think-aloud protocols and rational analysis requires domain expertise and are often time-consuming and error-prone [2]. Furthermore, hand-authored models can be too simplistic and are usually not verified or inconsistent with data.

Cognitive model discovery, sometimes called "KC model discovery" (in Educational Data Mining) or "Q matrix discovery" (in Psychometrics), has been attempted through a number of different methods, but the problem remains an open, important, and interesting one. Some attempts emphasize interpretability and application of the resulting models [3,4], while others have emphasized methods that minimize upfront human effort in feature engineering [5–9].

So far limited attention has focused on more ill-defined domains that are highly visual (e.g., classifying visual inputs) such as learning Chinese characters or non-discrete (probabilistic and/or with lots of exceptions) such as learning English grammar. In this paper, we tackle these ill-defined domains where complex prior perceptual skills and large amounts of background knowledge are required and where the input from the tutor is largely unstructured. We hypothesize that representations learned by machine learning techniques, which capture high-level features, can be used to create a cognitive model of human learning. We propose a novel architecture called *Cognitive Representation Learner* (CogRL) to automatically extract the set of KCs required for each problem in these domains using representation learning (i.e. transforming the raw data input to a representation that can be effectively exploited in machine learning tasks). CogRL does not require any student performance data and works directly on the unprocessed problem content in the tutor. Our contribution in this paper is two-fold: firstly, we show that CogRL architecture is able to identify accurate cognitive models which outperform the baselines in a wide variety of challenging domains such as RumbleBlocks, Chinese Character, and Article Selection. This is particularly useful in domains where a good human-authored cognitive model is unavailable or difficult to construct. Secondly, for domains where a cognitive model is available, we show that learned representations can be used to get accurate estimates of skill difficulty and learning rate parameters without using any student performance data.

2 Datasets

2.1 RumbleBlocks

Prior methods of cognitive model discovery only handle domains with textual problem content. We use data collected from the educational game *RumbleBlocks* [10] to test our system's ability to operate in a domain based on visual inputs. This game tasks students with building tower structures out of blocks in order to teach them basic structural stability and balance concepts. For the purposes of this study, we were less interested in modeling construction skills and more interested in modeling skills for recognizing when towers are more stable. To this end, we used a data set collected for a simplified task [11], where students were shown images of *RumbleBlocks* towers in a randomized order and are asked to classify each tower as either “concept 1” or “concept 2”. After each classification, students were provided with correctness feedback. The labels (concept 1 and concept 2) were intentionally kept vague so that students would be unable to use their prior stability and balance knowledge. The data set consists of twenty students classifying thirty towers.

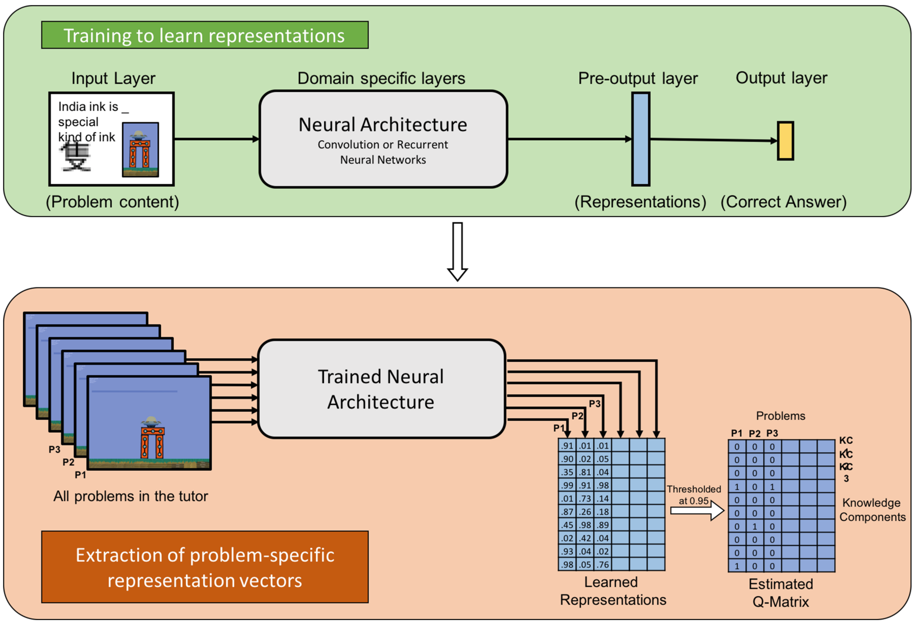


Fig. 1. Architecture of the Cognitive Representation Learner (CogRL) to estimate the Q-Matrix. For each domain, a neural architecture is chosen and trained using the problem content in the tutor and corresponding correct answer. The trained neural architecture is used to produce representation vectors in the pre-output layer for each problem. The resultant matrix is converted to a Q-Matrix by thresholding the values in the representations at 0.95.

2.2 Chinese Characters

For this domain, we use the Chinese vocabulary dataset[1] from the LearnLab Datashop [12]. The problems in the dataset contain 1105 unique Chinese characters with two types of responses, English and pinyin. Pinyin refers to the English character orthography for the Chinese character pronunciation. The dataset consists of 94 students and a total of 61,323 student-item transactions. We extract the set of Chinese characters in the dataset and convert them to 16×16 images for representation learning.

[1] https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=213.

2.3 Article Selection

For this domain, we use the data from English Article Selection task in the Intelligent Writing Tutor[2]. In this task, each question is a fill in the blank with three options: 'a', 'an' and 'the'. The dataset has 84 unique problems, 79 students with a total of 4,243 student-item transactions. The dataset also provides a human-authored cognitive model with 9 Knowledge Components.

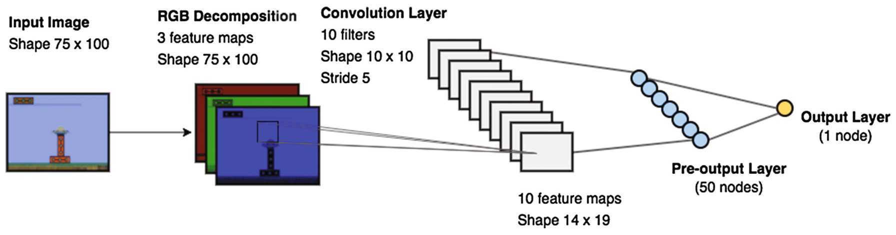


Fig. 2. Architecture of the convolutional neural network used for learning representations for the *RumbleBlocks* dataset. The pre-output layer (Fully Connected Layer) is used as the learned representation for each input image.

3 Methods

3.1 Cognitive Representation Learner

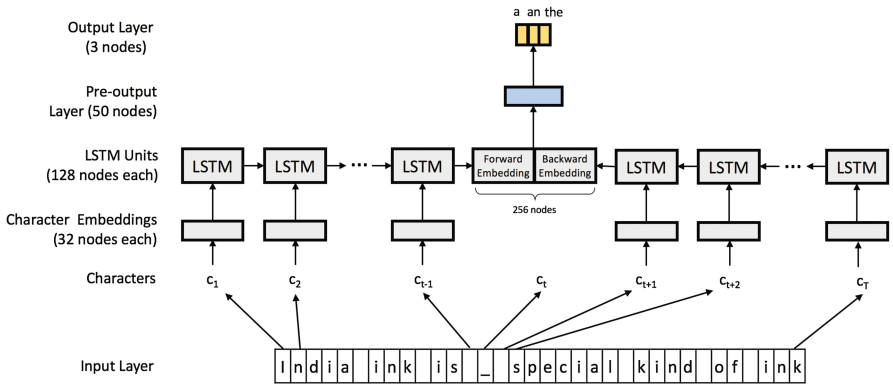
In this subsection, we describe the architecture of the proposed method, Cognitive Representation Learner (CogRL). For each domain, we train a Neural Architecture to predict the correct answer for the given problem in the domain. As shown in Fig. 1, the neural architecture, which is domain-specific, is connected to a fixed size pre-output layer, which will serve as the representations for corresponding problems. The pre-output layer is in turn connected to the output layer which predicts the correct answer for the given input problem. After training the architecture on the problems in the tutor, we use the trained model to compute the representations vectors in the pre-output layer for each problem. These representations are thresholded at 0.95 and used as columns of the estimated Q-matrix. In other words, each dimension of the learned representation constitutes a Knowledge Component in the predicted cognitive model. This cognitive model is evaluated by fitting an Additive Factors Model [13] using the student performance data.

In the *RumbleBlocks* and Chinese character datasets, the problems content is in the form of images. We use convolutional neural networks for these two datasets as they are shown to be effective in learning effective representations from pixel-based image data [14]. For the Articles Selection dataset, the challenge is that the size of the input is variable as opposed to fixed image size in the

[2] https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=307.

RumbleBlocks and Chinese character datasets. Convolutional Neural Networks can not handle variable input sizes. Recurrent models are suitable for handling variable length input by treating the input as a sequence. Particularly, we use a Long Short-Term Memory Network to learn representations for the Article Selection dataset.

In each of these architectures, we use the pre-output layer of the network as the representations for the corresponding input problem. The size of the representation or pre-output layer is not tuned to optimize the results and kept constant at 50 across all the architectures. These architectures and their training procedure are described in detail in the following subsections.



a an the
Output Layer (3 nodes)
Pre-output Layer (50 nodes)
LSTM Units (128 nodes each)
LSTM
Forward Embedding
Backward Embedding
256 nodes
Character Embeddings (32 nodes each)
Characters
c1 c2 ct-1 ct ct+1 ct+2 cT
Input Layer
I n d i a i n k i s _ s p e c i a l k i n d o f i n k

Fig. 3. Architecture of the Long Short-Term Memory (LSTM) network used for learning representations for the Article selection tutor. The pre-output layer (Fully Connected Layer) is used as the learned representation for each input sentence.

3.2 Convolution Neural Networks for RumbleBlocks and Chinese Characters

Convolution Neural Networks [14] are a type of feed-forward neural networks based on the convolution operation, which are typically used for processing visual input. Each convolutional layer consists of a set of learnable filters or kernels, which are convolved across the input image. The output is passed through a non-linear activation function, such as tanh, and scaled using a learnable parameter. Each element of y_j in the output of a convolution layer is calculated according to the following equation:

$$y_j = g_j \tanh\left(\sum_i k_{ij} * x_i\right) \qquad (1)$$

where x_i is the i^{th} channel of input, k_{ij} is the convolutional kernel, g_j is a learned scaling factor and $*$ denotes the discrete convolution operation calculated using the following equation:

$$(x * k)_{ij} = \sum_{p,q}^{R} x_{i-p,j-q} k_{p,q} \tag{2}$$

where R is the kernel size.

The architecture of the convolutional neural network used for learning representations for *RumbleBlocks* is shown in Fig. 2. The input image (75×100) is decomposed into red, green and blue channels which are connected to a convolutional layer consisting of 10 filters of size 10×10 with a stride of 5. The output of the convolutional layer is fully connected to a layer of 50 nodes, which in-turn is fully connected to the output layer predicting whether the configuration of rumble blocks in the input image is stable or not. The network is trained using stochastic gradient descent with a batch size of 32. After training the network, the value of the pre-output layer (50-dimensional) is used as the representation of the corresponding input image. We use the same architecture for Chinese Characters dataset except that the filter size is reduced to 4×4 with a stride 2 to match the size of a smaller 16×16 input image.

3.3 LSTMs for Article Selection

Long Short-Term Memory (LSTM) [15] networks are a type of Recurrent Neural Networks which are suitable for sequential data with a variable size of the input sequence. In addition to the input at the current time step, nodes in a recurrent layer also receive the output of the last time step as input. This recurrence relation makes the output dependent on all the inputs in the sequence seen till the current time step. In addition to the hidden state in a vanilla recurrent unit, LSTM units have an extra memory vector and they can use explicit gating mechanisms to read from, write to, or reset the memory vector. Mathematically, at each LSTM unit, the following computations are made:

$$\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hc}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t * c_{(t-1)} + i_t * g_t \\
h_t &= o_t * \tanh(c_t)
\end{aligned}$$

where h_t is the hidden state at time t, c_t is the cell state or memory vector at time t, x_t is the input at time t and i_t, f_t, g_t, o_t denote the input, forget, cell and out gates at time t, respectively.

The architecture used for learning representations for Article selection is shown in Fig. 3. The input question is split into two parts around the blank. Each character in both the parts has a 32-dimensional embedding. The part before the blank is fed into the forward part of the LSTM sequentially, while the part after the blank is fed into the backward LSTM in the reverse order. At the end of the sequence, both LSTM parts are flattened and combined to a layer of 256 neurons. This layer is fully connected to a pre-output layer with 50 neurons. This layer will serve as the representation for the given input question, which is fully connected to the output layer. The network is trained with all the questions in the English IWT dataset described in Sect. 2 using stochastic gradient descent with a batch size of 32. At the end of the training, for each input question, the pre-output layer embedding is stored as the feature representation of the question.

4 Experiments and Results

Each dimension of the representations learned using CogRL is considered to denote a Knowledge Component (KC). For each problem in a dataset, the representations are thresholded at 0.95, which means that if an element of the representation of a problem is greater than 0.95, then the problem is predicted to require the corresponding KC. This essentially creates a multi-KC Cognitive Model or a Q-Matrix whose rows are the thresholded representations for each problem. This automatically discovered cognitive model is evaluated by fitting the Additive Factors Model to the student performance data. We compare the CogRL cognitive model with two alternative theories of transfer of learning [16]. One, the Faculty Transfer model, is based on faculty theory of transfer that suggests that the mind is like a muscle and generally improves with more experience [17]. The other, the Identical Transfer model, is based on the identical elements theory of transfer that suggest learning transfer occurs across nearly identical stimuli [18]. These models are implemented as follows:

– Faculty Transfer: All the problems require a single common knowledge component.
– Identical Transfer: All the problems require a single unique knowledge component.

The proposed model is also compared with the best human expert cognitive model available with the tutoring system from which the data was collected. We use Item-stratified cross-validation Root Mean Square Error as the metric of comparison. The results shown in Table 1 indicate the CogRL cognitive model outperforms the baselines by a considerable margin, 0.444 vs 0.465 for Chinese Character, 0.449 vs 0.451 for Rumble Blocks and 0.399 vs 0.411 for Article Selection datasets. This indicates that the CogRL architecture is able to learn useful representations which constitute the underlying Knowledge Components for problems in various domains.

We also try to analyze the representation learned by CogRL qualitatively. Figure 4 shows two sets of problem images in the Rumble Blocks domain, which require a common knowledge component. Problem images which have a similar configuration of the blocks are predicted to require a common KC. Note that the exact position of the blocks in all the shown images is very different although they look similar visually.

Table 1. Cross-validated RMSE values for fitting Additive Factors Model (AFM) using various Cognitive Models on three different datasets. The proposed model, CogRL, outperforms the baselines by a considerable margin.

Dataset	Faculty Transfer	Identical Transfer	Best Human Model	CogRL
Chinese Character	0.471	0.493	0.465	**0.444**
Rumble Blocks	0.451	0.537	0.451	**0.449**
Article Selection	0.415	0.522	0.411	**0.399**

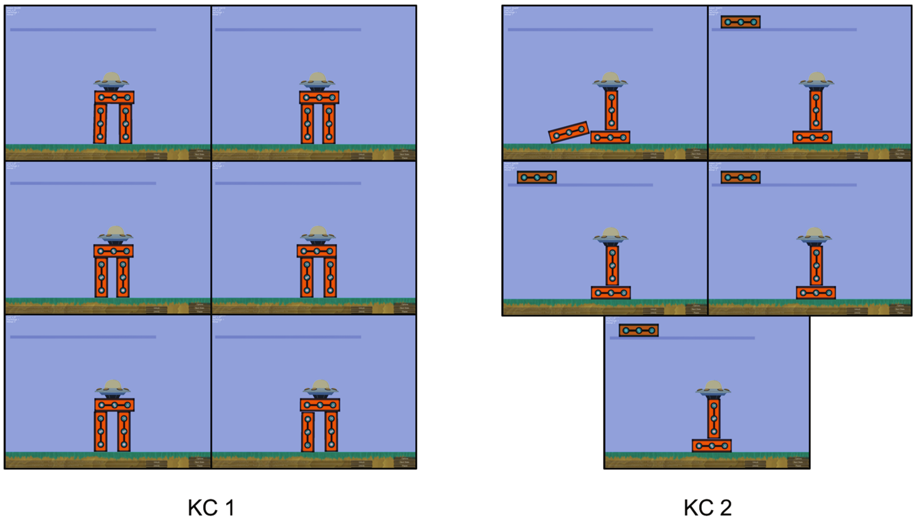


Fig. 4. Examples of non-identical **RumbleBlocks** problem images in two KCs discovered using Representation Learning. Input images which have a similar configuration of the blocks have a similar representation. Note that the images in one set are not identical, the exact position of the blocks is different in each image, although they look very similar visually.

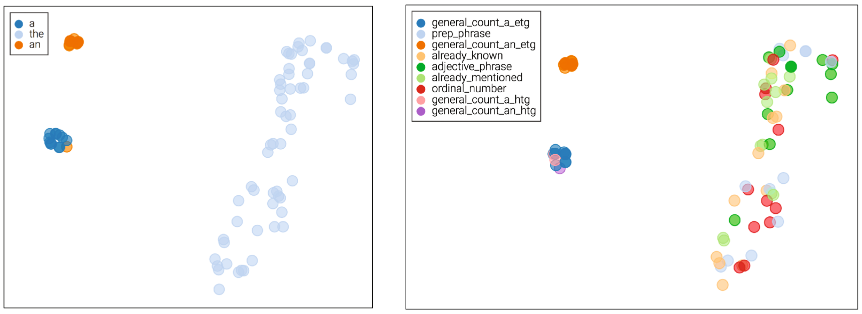


Fig. 5. t-SNE Visualization of Representations learnt for various Problems in the Article Selection dataset labelled using (a) Correct answer, (b) Underlying KC in the best human-authored cognitive model.

The representations learnt for the article selection dataset are visualized using t-Distributed Stochastic Neighboring Embedding (*t-SNE*) [19] in Fig. 5. t-SNE is a popular dimensionality reduction technique well suited for visualizing high dimensional data. The representations of problems are labeled according to the correct answer in Fig. 5(a) and according to the underlying KC in the human-authored cognitive model in Fig. 5(b). As shown in the Fig. 5(a), the representations for problems with the same answer are very similar to each other. However, one problem with answer 'an' is very similar to all problems with answer 'a' rather than other problems with answer 'an'. In Fig. 5(b), we can see that this problem is the only problem in the dataset with KC "general_count_an_htg". The problem is "The salesman is not ___ honest man", which belongs to this KC because the word following the blank starts with a vowel sound but not a vowel character. This makes it very similar to problems where the following word starts with a consonant and have 'a' as the correct answer. Many novice learners confuse problems in this KC to have 'a' as the correct answer. It is interesting to see that automatically learned representations also have this kind of relationship, which suggests that these representations might be indicative of human learning.

5 Extension: Estimating Skill Difficulty and Learning Rates

Intelligent Tutoring Systems are able to improve student learning across a wide range of domains by utilizing student modeling techniques (such as Additive Factors Model [13], Bayesian Knowledge Tracing [20], Performance Factor Analysis [21]) to track the skills students have acquired and to focus practice on unlearned skills. However, student modeling approaches require reasonable initial parameters in order to effectively track skill learning. In prior work, researchers have used pilot studies with fixed, non-adaptive, curriculum to empirically determine the difficulty and learning rates of skills in order to appropriately set the knowledge tracing parameters. In the previous sections, we showed that representation learning using neural architectures can be used for automatic cognitive model discovery, which is especially effective in domains where a good human-authored cognitive model is unavailable. In this section, we show that in domains where a good cognitive model is available (such as article selection), representation learning can be used to estimate the difficulty and learning rates of skills or knowledge components in the given cognitive model. For this task, we leverage the formalism of the Apprentice Learning Architecture [22] to simulate entire classroom studies for Article Selection dataset and demonstrate that empirical estimates of skill difficulty and learning rate parameters from these simulation data have high agreement with the parameters empirically estimated from human data. It is not possible to study the other domains in this context as a human-authored cognitive model is unavailable. The Apprentice learner is trained using the same sequence of problems as received by students in the Articles selection dataset. For each problem, the representations learned by CogRL are passed as input features to the Apprentice Learner. The learner fits a decision tree classifier on seen problem examples to simulate learning. The data generated using this simulation is fit using Additive Factor Model to get skill slope and intercept estimates. These estimates are compared to parameter estimates using the original student performance data.

As a baseline, we also train the Apprentice Learner using human-authored features defined by domain experts. For the article selection tutor, the domain experts defined 6 binary features, each of which is true if the following conditions hold true:

– 'next_word_starts_with_vowel': Whether the word following the blank starts with a vowel. This feature approximates whether the noun following the article begins with a vowel sound.
– 'next_word_ending_st_nd_rd_th': Whether the word following the blank ends with 'st', 'nd', 'rd' or 'th'. This feature is an approximation of whether the next word is an ordinal number or not.
– 'contains_that_where_who': Whether the question contains 'that', 'where' or 'who'. This feature is an approximation of whether the noun following the article is made definite by a prepositional or an adjective phrase.
– 'next_word_already_mentioned': Whether the word following the blank is already mentioned elsewhere in the question. The feature is an approximation of whether noun that follows the article is already known or mentioned.
– 'next_word_ends_in_s': Whether the word after the blank ends with a 's'. This feature approximates if the noun following the article is not a singular count noun.
– 'contains_but_comma': Whether the sentence contains 'but' or ','. This feature approximates whether the question has two clauses and the noun following the article is referred in the first clause and therefore, already known. For example, "When I have watermelon, I try not to eat ___ seeds".

Note that the article selection task is fairly complex and it is extremely difficult to author text-based features that are sufficient to answer all questions in the dataset correctly. For example, authoring text-based features which recognize vowel sound in words not starting with vowels like 'honest' is extremely difficult. The features authored by the experts are shallow and are sufficient to answer only 75% of the questions correctly.

5.1 Parameter Estimation Results

The parameter estimates for the Article Selection dataset using (1) the original student performance data, (2) using the simulated data with Human-authored features, and (3) using the simulated data with CogRL features are shown in Table 2. As shown in the table, the parameter estimates using CogRL features have a high correlation of 0.986 for slope, and 0.748 for intercept with the parameter estimates using the original data. This is considerably higher than the correlation of parameter estimates using Human-authored features. Most of the human-authored features are deep and result in very fast learning rate

Table 2. Table showing parameter estimates for the Article Selection dataset using the original student performance data, using the simulated data with Human-authored features, and using the simulated data with Representation Learning features. The parameter estimates using representation learning correlate highly with the parameter estimates using the original data.

	Original data		Apprentice Learner trained using			
			Human-Authored features		CogRL features	
KC Name	Intercept	Slope	Intercept	Slope	Intercept	Slope
produce_adjective_phrase	0.818	0.093	0.881	0	0.563	0.156
produce_already_known	0.768	0.092	0.547	1.208	0.332	0.202
produce_already_mentioned	0.930	0.044	0.914	16.803	0.721	0.048
produce_general_count_a_etg	0.604	0.066	0.821	0	0.396	0.022
produce_general_count_a_htg	0.202	0.660	0.119	2.627	0.065	2.221
produce_general_count_an_etg	0.670	0.031	0.14	0.882	0.044	0.205
produce_general_count_an_htg	0.467	0.817	0	0	0.037	3.208
produce_ordinal_number	0.783	0.151	0.958	0	0.814	0.003
produce_prep_phrase	0.660	0.138	0.58	0.082	0.348	0.331
Correlation with Original			0.742	–0.187	0.748	0.986

for certain KCs such as 'already_mentioned' and 'already_known', while since they don't cover all features required for learning some other KCs such as 'general_count_an_htg', they have a learning rate of 0. CogRL captures features essential for learning all KCs and seems to better model the struggles that learners are experiencing to acquire deep features. While it would be very difficult to train the Apprentice Learner from raw problem data, CogRL features are able to constitute the right amount of prior knowledge necessary to simulate learning in this domain. Apart from providing accurate parameter estimates without using any student performance data, the CogRL framework also minimizes the amount of human-authoring necessary to conduct simulation studies in this challenging domain.

6 Related Work

There has been a lot of interest in automating cognitive model discovery in the recent past. Learning Factors Analysis is a method for cognitive model evaluation and improvement which semi-automatically refines a given skill set. The improved cognitive model discovered using LFA has been used to redesign an intelligent tutoring system and shown to improve learning gains [4]. However, LFA requires human-provided factors that require some knowledge engineering or cognitive task analysis effort. eEPIPHANY [6] attempts to overcome this limitation by using a collection of data-mining techniques to more automatically improve a human-crafted set of skills. LFA and ePHIPHANY both require a human-crafted set of skills as well as student performance data for cognitive model discovery and improvement.

The requirement of student performance data makes these methods unusable for authoring a cognitive model for a new domain or a tutor with new problems. Li et al. [8] is notable prior attempt at learning a cognitive model without student performance data. Their SimStudent learns a cognitive model by being tutored in the domain through demonstrations of correct actions and yes-no feedback on SimStudent attempts at actions. They show improved cognitive models in various domains such as algebra, stoichiometry and fraction addition [2]. Furthermore, the skill learning in SimStudent was also integrated with feature learning using probabilistic Context Free Grammars (pCFG) to automatically learn features to train the SimStudent [23]. However, as discussed previously, SimStudent requires structured input from the tutor interface and the learning method is mostly applicable in well-defined problem domains where minimal background knowledge is required.

Among approaches using neural networks in educational data mining, Wang et al. [9] train an LSTM to predict student's learning over time using student performance data in programming exercises. The t-SNE visualization of the hidden layer outputs of their trained neural network shows clusters of trajectories sharing some high-level properties of the programming exercise. Pardos and Dadu [24] use contextual representations learnt by a skip-gram model to predict missing skill from a KC model. Michalenko et al. [25] use word embeddings detect misconceptions from students' responses to open-response questions.

In contrast to prior work, we tackle domains where the tutor interface is largely unstructured and problem solving requires complex prior perceptual skills and large amounts of background knowledge. Furthermore, while handling these complex domains, our methods do not require any student performance data which makes them suitable for initializing cognitive models while designing a new tutor. We also provide a method for estimating skill difficulty and learning rate without any student performance data using simulations of Apprentice learner. These estimates can be used as initialization in new tutors to provide a better estimate of mastery for each student.

7 Conclusion and Future Work

We showed that representation learning using neural architectures can be used for automatic cognitive model discovery without using any student performance data, which is especially effective in domains where a good human-authored cognitive model is unavailable or authoring a good cognitive model is difficult. Qualitative analysis of representations learnt by CogRL suggests similarities with human learning. For domains where a cognitive model is available, we show that representation learning can be used to get effective estimates of skill difficulty and learning rate parameters without using any student performance data. In future, the CogRL framework can be modified to make the representations more interpretable and provide constructive feedback for improving instructional design.

References

1. VanLehn, K., Jordan, P., Litman, D.: Developing pedagogically effective tutorial dialogue tactics: experiments and a testbed. In: Workshop on Speech and Language Technology in Education (2007)
2. Li, N., Stampfer, E., Cohen, W., Koedinger, K.: General and efficient cognitive model discovery using a simulated student. In: Proceedings of the Annual Meeting of the Cognitive Science Society, vol. 35 (2013)
3. Cen, H., Koedinger, K., Junker, B.: Learning factors analysis – a general method for cognitive model evaluation and improvement. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) ITS 2006. LNCS, vol. 4053, pp. 164–175. Springer, Heidelberg (2006). https://doi.org/10.1007/11774303_17
4. Koedinger, K.R., Stamper, J.C., McLaughlin, E.A., Nixon, T.: Using data-driven discovery of better student models to improve student learning. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS (LNAI), vol. 7926, pp. 421–430. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39112-5_43
5. González-Brenes, J., Mostow, J.: Dynamic cognitive tracing: towards unified discovery of student and cognitive models. In: Proceedings of the 5th International Conference on Educational Data Mining (2012)
6. Matsuda, N., Furukawa, T., Bier, N., Faloutsos, C.: Machine beats experts: automatic discovery of skill models for data-driven online course refinement. Int. Educ. Data Min. Soc. (2015)
7. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Deep knowledge tracing. In: Advances in Neural Information Processing Systems, pp. 505–513 (2015)
8. Li, N., Cohen, W., Koedinger, K.R., Matsuda, N.: A machine learning approach for automatic student model discovery. In: Proceedings of the 4th International Conference on Educational Data Mining (2011)
9. Wang, L., Sy, A., Liu, L., Piech, C.: Learning to represent student knowledge on programming exercises using deep learning. In: Proceedings of the 10th International Conference on Educational Data Mining (2017)
10. Christel, M.G., Stevens, S.M., Maher, B.S., Brice, S., Champer, M., Jayapalan, L., Chen, Q., Jin, J., Hausmann, D., Bastida, N., et al.: Rumbleblocks: teaching science concepts to young children through a unity game. In: 2012 17th International Conference on Computer Games (CGAMES), pp. 162–166. IEEE (2012)
11. MacLellan, C., Harpstead, E., Aleven, V., Koedinger, K.: Trestle: a model of concept formation in structured domains. Adv. Cogn. Syst. **4**, 131–150 (2016)
12. Koedinger, K.R., Baker, R.S., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J.: A data repository for the edm community: the pslc datashop. In: Handbook of Educational Data Mining, vol. 43, pp. 43–56. CRC Press, Boca Raton (2010)
13. Cen, H.: Generalized learning factors analysis: improving cognitive models with machine learning. ProQuest (2009)
14. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
16. Koedinger, K.R., Yudelson, M.V., Pavlik, P.I.: Testing theories of transfer using error rate learning curves. Top. Cogn. Sci. **8**(3), 589–609 (2016)

17. Nichols, R., Yaffe, G., Reid, T., Zalta, E.N. (eds.): The Stanford Encyclopedia of Philosophy. Winter 2016 edn. Metaphysics Research Lab, Stanford University (2016)
18. Thorndike, E.L.: The Principles of Teaching: Based on Psychology, vol. 32. Routledge, Oxford (2013)
19. Maaten, L.: Learning a parametric embedding by preserving local structure. In: van Dyk, D., Welling, M. (eds.) Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics. Volume 5 of Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA. PMLR, pp. 384–391, 16–18 April 2009
20. Corbett, A.T., Anderson, J.R.: Knowledge tracing: modeling the acquisition of procedural knowledge. User Model. User-Adap. Inter. **4**(4), 253–278 (1994)
21. Pavlik, P.I., Cen, H., Koedinger, K.R.: Performance factors analysis -a new alternative to knowledge tracing. In: Proceedings of the 2009 Conference on Artificial Intelligence in Education: Building Learning Systems That Care: From Knowledge Representation to Affective Modelling, pp. 531–538. IOS Press, Amsterdam (2009)
22. MacLellan, C.J., Harpstead, E., Patel, R., Koedinger, K.R.: The apprentice learner architecture: closing the loop between learning theory and educational data. In: Proceedings of the 9th International Conference on Educational Data Mining (2016)
23. Li, N., Matsuda, N., Cohen, W.W., Koedinger, K.R.: Integrating representation learning and skill learning in a human-like intelligent agent. Artif. Intell. **219**, 67–91 (2015)
24. Pardos, Z.A., Dadu, A.: Imputing kcs with representations of problem content and context. In: Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, pp. 148–155. ACM (2017)
25. Michalenko, J.J., Lan, A.S., Baraniuk, R.G.: Data-mining textual responses to uncover misconception patterns. stat. **1050**, 30 (2017)