An Interactive Conversational Agent to Aid Human Learning

Bradley Whitehall*
Carnegie Mellon University
Pittsburgh, PA 15213
bwhiteha@andrew.cmu.edu

Deigant Yadava*
Carnegie Mellon University
Pittsburgh, PA 15213
dyadava@andrew.cmu.edu

Dheeraj Mohandas Pai*
Carnegie Mellon University
Pittsburgh, PA 15213
dmohanda@andrew.cmu.edu

Vinay Nair*
Carnegie Mellon University
Pittsburgh, PA 15213
vinayn@andrew.cmu.edu

Abstract

In this project, we introduce an intelligent agent that can help humans learn a new subject by intelligently scheduling the introduction of new concepts from that subject to the learner. We explore the application of this agent on the task of language learning wherein we aim to help a learner improve their command on a new language. The agent keeps track of the current knowledge and capability of the learner and intelligently selects exercises for the user based on this current knowledge state. Our work aims to provide a method to deduce the knowledge state of a user using the user's task history, such as past mistakes and results from tests. Our method involves a knowledge tracing model combined with a reinforcement learning agent to select the best possible next exercise for the user. The tested model significantly outperforms a random policy, which was selected as our baseline comparison. These successfully results demonstrate the capabilities of reinforcement learning as a viable technique for online learning modules. We have made all code and pre-trained models from this research public to encourage further research and development in this domain.[1].

Keywords: Reinforcement learning, Deep knowledge tracing, Conversational AI

1 Introduction

One of the best ways to learn any subject or topic is to be incrementally exposed to different concepts of the subject over time. This iterative learning needs to be done in such a way that the concept introduced at every step is of a slightly higher complexity as compared to the concepts that the learner already knows. This means that the learning process can be enhanced for a learner if the concepts that are introduced in each step are scheduled according to the current knowledge state of the learner. The same is true if a learner is trying to learn a new language. One of the most effective ways to learn a new language is through immersion, but not everyone has access to language immersion programs. We plan to tackle this by developing an intelligent agent that can carry out personalized conversations with the learner.

This project would have two major components:

[1] Github Repository: https://github.com/deigant1998/IntroToDeepLearning11785Project , *equal contribution

Knowledge tracing: At every step, in order to personalize the conversations, the agent will keep track of the word-level mistakes that the learner makes and use this data to generate a representation of the knowledge state of the user (conversely, using the representation of the knowledge state we can predict the probability of a user making a mistake on a particular word in a sentence). This knowledge representation is used by the agent to generate relevant conversation.

Conversation Agent: The conversational agent is responsible for generating new conversations with the learner based on the current knowledge state of the user. We explore the applications of Sequence to Sequence models to generate the next dialogue for the conversation (where the previous reply from the user and the knowledge state of the user can be fed as the input to the model). We train the agent using Reinforcement learning algorithms (Q-learning) to find the best action to select a conversation that can help in improving the knowledge of the user. The reward for the reinforcement learning algorithm is based on the improvement in the knowledge state of the user after having a conversation with the agent.

Our contributions are summarized as follows:

- We propose a novel architecture based on Reinforcement learning and Deep knowledge tracing that can help humans in language learning.
- We explore novel previously unexplored architectures, especially based on attention, for Deep knowledge tracing in the task of second language acquisition. In particular, we focus on architectures that can be easily integrated with our RL agent.
- We train and evaluate a Q-learning based RL agent to intelligently schedule conversations that outperforms the baseline in terms of both cumulative reward and sample efficiency.

2 Related Work

2.1 Knowledge Tracing (KT):

This approach involves the use of knowledge tracing for understanding what the user learned based on their conversations with the bot. Prior works in Recurrent Neural Network(RNN) based Deep knowledge tracing (DKT) [20], Transformer based DKT [18], GNN based DKTs [15, 26] show that deep knowledge tracing can be used effectively to model the current knowledge state of a student purely based on the performance of previous questions and the time taken to solve them. Given a new set of questions the DKT models predict if the student will be able to solve the question or not. Though these works are not focused on language learning, the approach can be used on language learning based on the dataset available from Duolingo.

Further, a Knowledge Graph based approach incorporating autoencoders to compute recommendations has been developed by Bellini et al. [2]. The authors make use of a Knowledge Graph (KG) and build an autoencoder to estimate this graph which in turn tries to learn the user's learning patterns. Although this is not related to education, the approach ties really closely with what we're planning to achieve.

To improve the capability of knowledge graph completion tasks Zhang et al. introduced a new model architecture for relation graph neural networks [31]. Their model–Relational Graph neural network with Hierarchical ATtention (RGHAT)–leverages information from local relationships in the knowledge graph. The use of attention to maintain neighborhood information allows the model to notably outperform previous state of the art models in knowledge graph completion tasks.

An additional method for improved knowledge tracing was implemented by [17], which involves the ensemble of multiple independent knowledge tracing models [17]. This method has shown to provide knowledge tracing with greater accuracy than any single knowledge tracing model has been able to achieve. One of the individual models employed by [17] is a gradient boosted decision tree using the light GBM framework. The light GBM framework was developed by Ke et al. [8]. The light GBM framework allows for training a decision tree using only a fraction of the dataset at a time. This makes it an ideal framework to train a decision tree on a large, sparse data set such as ours.

A Bayesian knowledge tracing (BKTs) [3] and Individualized BKT (I-BKT) [30] models can also be used to simulate the learners understanding of a subject. Though DKTs outperform BKTs in knowledge tracing, BKTs can often be used when the number of samples is not sufficient to train a

deep learning model. Bassen et. al. [1] uses BKT models to simulate student's understanding of the subject based on minimal test scores and samples. This is very useful when there is a human in a loop - where data is both online and real-time provided by the student using the learning app.

2.2 Conversational Agent

As a motivation for this research we use the work done by Ruan et al. [21] where they show that leveraging interactive chatbots can help in improving the language learning experience of students. The authors worked with 56 Chinese students who tried to learn English as a second language and showed that their performances improved more when using an interactive chatbot (as compared to traditional listen and repeat methods) when evaluated by the IELTS grading standard for voluntary learning.

Recent work by Bassen et al. [1] shows that reinforcement learning can be used to effectively schedule the learning activities of a person. The RL agent is trained to suggest the educational activities to maximise the total gain in knowledge and minimise total time consumed while in learning the subject. The policy is optimized using sample-efficient proximal policy optimization (PPO) which is very essential in the case where the number of samples from the users would be significantly low.

A sample efficient policy optimization is very essential in our case where the number of samples from the users would be significantly low. Model based re-inforcement learning have showed promising results for sample-efficiency without much compromise on the performance [28]. When it comes to policy optimizations Proximal Policy optimization(PPO) [24], and Trust region policy optimization (TRPO) [23], Deep deterministic Policy gradients [13](DDPG), Twin delayed DDPG [6], have shown much better sample efficiency compared to the basic policy gradient optimizations.

3 Model Architecture

3.1 Overall Vision

The overall vision that we have in mind for this project is to create an interactive agent that can converse with the users (either by asking questions or just generally discussing a topic) in a target language (which the users want to learn). The conversations that the agent generates should be tailored to the skill level of each user. In order to do that same, the agent needs to be able to have the following two components:

1. A knowledge tracing module that will keep track of the user replies and build a representation of the current skill level of the user
2. A conversation generator that will select the next sentence/unit of conversation on the basis of the current skill level of the user.

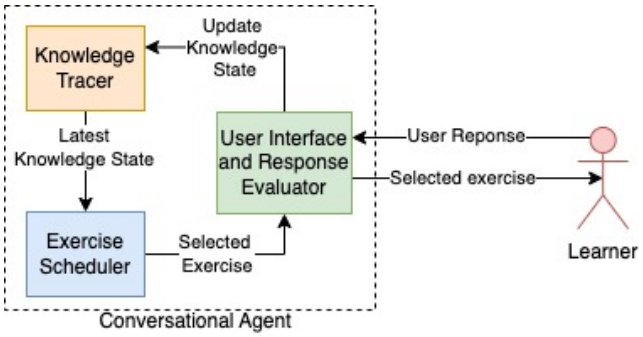


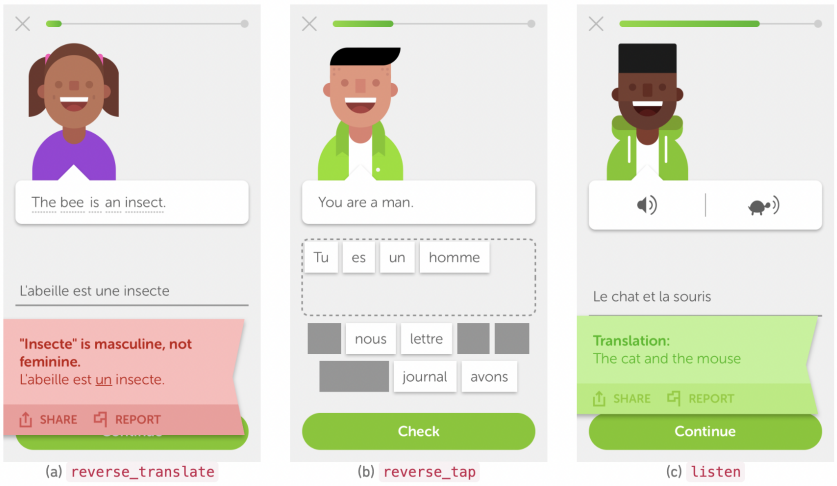
Figure 1: End to End view of the conversational agent

3.2 Dataset Description

For the specific task of knowledge tracing and conversation generation targeted to language learning, we make use of the dataset for Second Language Acquisition modeling that was released by Duolingo

[25]. For every user, the dataset consists of the history of interactions of the user (in terms of the exercises that have been completed) :

1. Each exercise can from one of the following formats:
   (a) reverse_translate: Given a language in the known language, translate it to the new language (being learned)
   (b) reverse_tap: Given a language in the known language, translate it to the new language (being learned) by selecting options from a given set of tags.
   (c) listen: Listen to a sentence in the language being learned and then transcribe it.



(a) reverse_translate (b) reverse_tap (c) listen

Figure 2: Exercise formats available in Duolingo[25].

2. For each exercise, we are given the following:
   (a) The written prompt that was provided to the user in the known language (if the format was reverse_translate or reverse_tap).
   (b) The correct answer for the exercise (closest to the answer that was provided by the user).Due to the existence of synonyms, homophones, and ambiguities in number, tense, formality, etc., each exercise on Duolingo may have thousands of correct answers. Therefore, to match the student's response to the most appropriate correct answer from the extensive set of acceptable answers, Duolingo utilizes FSTs.
   (c) Labels for each token in the correct answer to indicate if the user got that token right or not
   (d) Morhopology labels for each token in the closest correct answer
   (e) Time taken by the user to complete the exercise.
   (f) Time since the user started learning that language

We will be using this dataset to train our knowledge tracing model as well as create the initial version of the conversation generator.

```
# prompt:Yo soy un niño.
# user:XEinXf5+  countries:CO  days:0.003  client:web  session:lesson  format:reverse_translate  time:9
DRihrVmh0101  I            PRON    Case=Nom|Number=Sing|Person=1|PronType=Prs|fPOS=PRON++PRP              nsubj       4  0
DRihrVmh0102  am           VERB    Mood=Ind|Number=Sing|Person=1|Tense=Pres|VerbForm=Fin|fPOS=VERB++VBP   cop         4  0
DRihrVmh0103  a            DET     Definite=Ind|PronType=Art|fPOS=DET++DT                                det         4  0
DRihrVmh0104  boy          NOUN    Number=Sing|fPOS=NOUN++NN                                             ROOT        0  0
```

Figure 3: Example Exercise in Duolingo

3.3 Knowledge Tracing Module

The expectation of the knowledge tracing model is as follows:

1. Given the history of the user, i.e. the set of exercises that have been completed by the user (along with the labels indicating whether or not the user got a particular token in the exercise correct) : the model should be able to predict the performance of the user on a new exercise
2. If the set of exercises that have been completed by the user along with the labels for each token in each exercise is given by E = $\{\{E_1, L_1\}, \{E_2, L_2\}, \{E_3, L_3\}, ...\}\}$ and the new exercise that we are going to predict on is given by $X = \{X_1, X_2, X_3 \dots X_n\}$, then the goal of the knowledge module is to predict the following: $P(Y_i|X, E)\forall i \in \{1, 2, ...n\}$ where Y_i refers to the probability that the user will get the token X_i incorrect.

In order to train our knowledge model, we make use of the tokens in the closest matching correct answer that has been provided in the dataset for each exercise. This is because the tokens corresponding to the correct answer actually represent the concepts that we want the user to learn (i.e. the concepts from the new language). We make use of the GloVe embeddings of these tokens in-order to make sure that we capture the meaning/essence of each of these tokens.

3.3.1 Proposed Architecture:

We propose to make use of an encoder-decoder architecture to develop our knowledge model. The following figure provides a high level overview of the architecture of the knowledge model:

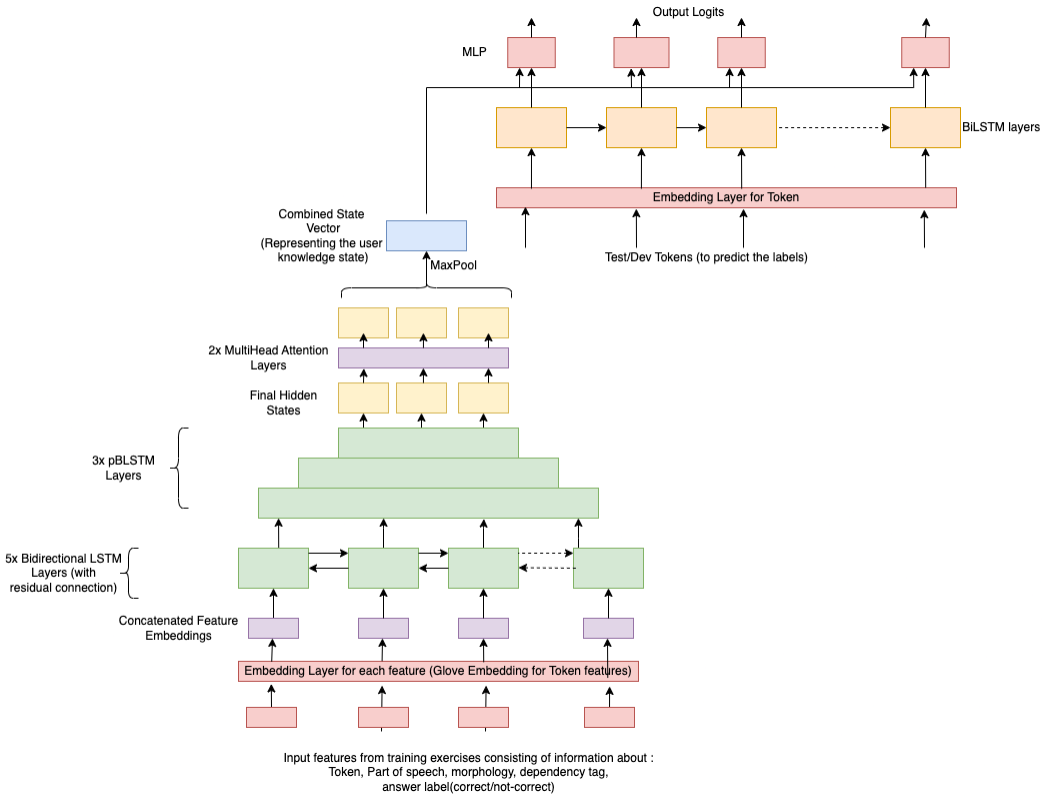


Figure 4: Encoder Decoder Architecture for Knowledge tracing

In our knowledge model, the encoder is responsible for creating a hidden representation of the knowledge state of the user and the decoder will be responsible for making use of that knowledge state and predicting the performance of the user on a new unknown set of exercises.

The overall architecture of the knowledge model can be summarized as follows:

1. The encoder consists of 5 layers of Bidirectional LSTM, followed by 3 pBLSTM layers to create a condensed representation of the input (user history).
2. We make use of 2 multi head attention layers on the outputs of the pBLSTM to take any long range dependencies between the inputs into consideration.
3. The output from the multi head attention layers is condensed into a single vector by making use of max-pooling. This single vector is a representation of the knowledege state of the user.
4. The decoder again, consists of 5 layers of Bidirectional LSTM to create a hidden representation of the test exercises (on which we want to predict the responses of the user). The decoder will create a single hidden representation for each token in the test exercises.
5. Finally, the decoder has a 3 layered Feed Forward network that takes in the hidden state (of the test token) along with the knowledge state of the user and then predicts the probability that the user will get that particular token incorrect.

Please refer to section 4 for a detailed explanation of the training of the knowledge module.

3.4 Conversational Agent

The goal of the conversational agent is to generate the next exercise that should be provided to the user such that the expected knowledge gain of the user is maximized. We followed an iterative approach to develop a Reinforcement Learning based agent that will generate the next exercise.

3.4.1 Scheduling Algorithm for exercise selection

As the first step towards developing an interactive agent, we first developed a simple Reinforcement Learning based scheduling algorithm. The Agent in this case is responsible for choosing the best possible way to schedule the existing exercises from the Duolingo dataset such that the knowledge gain of the user is maximised.

In our implementation the reinforcement learning agent is trained using Q-learning. The following are some architectural details of the reinforcement learning agent:

1. The agent is actually implemented as a simple linear regression model that is responsible for modelling the Q-Value for any given (state, action) pair.
2. In our case, the state is actually the knowledge state vector of the user and the action space is the set of exercises that can be presented to the user. During training, the input to the RL agent consists of the vector representing the knowledge state of the user and the embedding of a candidate exercise which are computed using Sentence BERT.
3. We follow an ϵ greedy policy to train the reinforcement learning agent. In every iteration, the reinforcement learning selects a random action/exercise with probability ϵ and with probability $1-\epsilon$, the agent iterates over the entire set of candidate exercises that are available, computes the Q value for each of them and then selects the one that has the maximum Q-value.

Please refer to section 4 for a more detailed explanation of the training procedure used for the reinforcement learning agent.

3.4.2 Reinforcement Learning and Simulated Learners

In the ideal scenario, the Reinforcement Learning algorithm (as mentioned above) would have to be trained on the outputs from human language learners. In the absence of such data, we will be making use of Simulated learners based on Bayesian Knowledge Tracing (BKT)[3] Bayesian knowledge tracing can be used to model the learning curves for the skills of each user. As per the original proposal in Bayesian Knowledge tracing, each "skill" is either in the learned state or the unlearned state. The following probabilities can be used to model the growth of a skill for a particular user:

1. $P(L_0)$: The probability that the skill is in the learned state for the user at time 0.

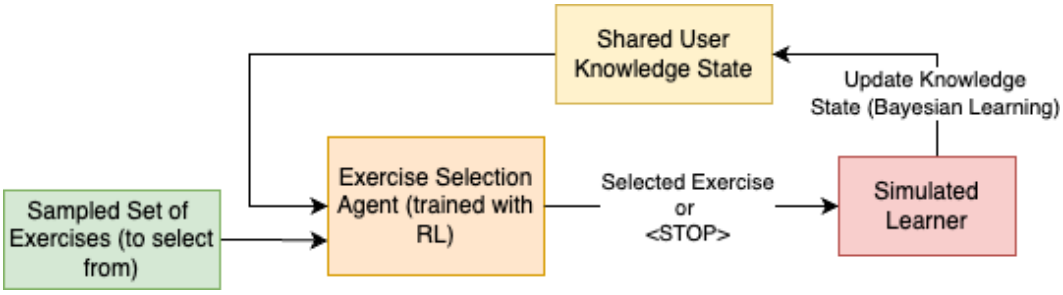


Figure 5: An RL based scheduling Agent

2. $P(T)$: The probability that the skill will transition from the Unlearned state to the learned state when a chance to apply the given skill is presented to the user.
3. $P(G)$: The probability that the user will guess the correct answer to an exercise which requires the application of a skill that is currently in the unlearned state for the user
4. $P(S)$: The probability that the user will slip and provide the incorrect answer to an exercise which requires the application of a skill that is currently in the learned state for the user
5. $P(L_t)$: The probability that the skill is in the learned state for the user at time t.

Using the above probabilities, the probability that the user will give the correct answer to an exercise at time 't' is given as:

$$P(C) = P(L_t) * (1 - P(S)) + (1 - P(L_t)) * P(G)$$

and once the user has provide the answer to an exercise, the probability that the skill will be in the learned state at time t+1 will be given as:

$$P(L_{t+1}) = P(L_t|obs_t) + (1 - P(L_T|obs_t)) * P(T)$$

where $P(L_t|obs)$ refers to the posterior probability of the skill being in the learned state once we have seen the response (obs_t) of the user to the exercise at time 't' and

$$P(L_t|obs_t = correct) = \frac{P(L_t) * (1 - P(S))}{P(L_t)(1 - P(S)) + (1 - P(L_t))P(G)}$$

$$P(L_t|obs_t = incorrect) = \frac{P(L_t)(P(S))}{P(L_t)P(S) + (1 - P(L_t))(1 - P(G))}$$

In our case, we will make of a Simulated learner to test the policy that is being created by the Reinforcement agent and provide a reward accordingly. For our application, the Simulated learner (based on BKT) has been implmented using the following assumptions:

1. In order to correctly provide the answer to a particular token in an exercise (eg: provide the correct translation of a particular word), the user needs to have a complete understanding of the token itself along with the corresponding part speech , morphological features and dependency structure of the token.
2. Each of the possible values of the tokens, part of speech, morphological features, dependency structure are modelled as a skill in the BKT learner.
3. Once the user is presented with an exercise, first the response of the user to each token in the exercise is computed based on the present skill state of the user. Once we have the generated the answers to each token (which in this case would be whether or not the user provides the correct answer to a particular token), we update the skills associated with each token on the basis of the observed value.

3.4.3 Integrating a Conversation Generator

Once we had the core components (i.e. the Reinforcement Learning Agent and Knowledge tracing model) in place, we incorporated a conversational element to language learning. At this step instead of using a shared knowledge state between the simulated learner and the Reinforcement learning agent, we integrated the encoder-decoder based knowledge tracing model with the Reinforcement Learning based exercise scheduling agent. This allows the interactive agent to build a representation of the current skill set of the user over time and select the appropriate exercises. Once the RL agent has selected a particular exercise, it is fed through a Conversation Generator (that will be a Sequence to Sequence Model) that will be trained to convert the given simple exercise into a more conversational question. For eg. For a user learning English (using Spanish as the base language), if the RL agent indicates that the exercise with the corresponding English translation as "I am not perfect" is the best, the conversational agent can generate the sentence as "Can you please translate 'Yo no soy perfecto' into english for me"?

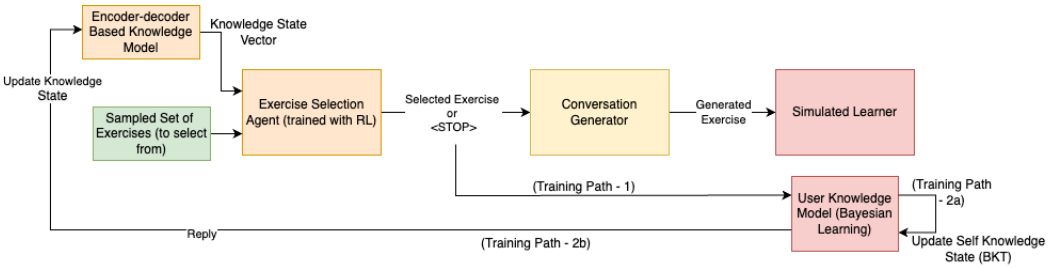


Figure 6: End to End integration of the Conversation Generator with the scheduling Algorithm

The conversation generator used is a Sequence-2-Sequence, pre-trained Google T5 model, that is fine-tuned on the prompts and the corresponding correct answers in the Duolingo dataset.

The overall training approach for the RL agent along with the encoder-decoder based knowledge tracing model will be the same as that mentioned in the previous section. The encoder-decoder based knowledge tracing model will be pre-trained and will be frozen while training the RL agent - it will only be used to generate the knowledge state vectors on the basis of the responses that are provided by the simulated learners.

4 Implementation Details

4.1 Training Details for the Knowledge Tracing Module

The knowledge model makes use of the following end-to-end algorithm for training:

1. For a given user, we consider a randomly sampled sub-sequences of n = '256' exercises and divide it into 2 parts each with n/2 exercises. The first n/2 exercises (say set I) are used to represent the current history of the user and the rest of them (say set T) represent the exercise for which we want to predict the outcome. For each token in the set I, we extract the following features:
   (a) The embeddings for each token (we make use of GloVe embedding)
   (b) We project each of the other categorical features such as (part of speech tag, dependency label of the token (in the semantic parsing of the exercise) and the label of the token (i.e. whether the user got the token correct or not)) into an embedding vector.
   (c) All the above mentioned features are concatenated to form an input for the encoder.
   (d) The encoder hidden states from all timestamps are be condensed into a single vector using Max Pooling. This vector will represent the knowledge state of the user because it has information about the exercises that have been attempted by the user as well as the labels for each token in that exercise (i.e the information about whether the user got the token correct or not).

(e) Now, once we have a condensed representation of the knowledge state of the user, we make use of this in the decoder to predict the performance of the user on a test set of exercises (i.e. the second subset T). The input to the decoder will be similar to the encoder except that the value of the 'label' for the exercise will be provided as 'unknown' (represented by the number '2' in the input since the values 0 and 1 represent the label for correct and incorrect responses respectively). The BiLSTM layers in the decoder will create a hidden representation for each token in the test set. This hidden representation at each time-step (i.e. for every token) is concatenated with the knowledge state representation (obtained from the encoder) and is then passed through an MLP network to get the final probability of the getting the token wrong.

(f) The network is trained using the ground truth labels, for the test exercises, that are available to us in the training data.

During inference, we make use of the history of the user, that is available to us, to create the knowledge state vector (i.e. just make use of the encoder part of the network) since we are only interested in the representation of the knowledge state. (Note: The knowledge state of the user will be updated every time we get a response from the user for an exercise i.e. at every step of the interaction between our agent and the user. This is because the history of interactions of the user will be updated every time we provide a new exercise to the user and the user attempts it).

4.2 Training Details for the Reinforcement Learning Agent

The training of the RL agent is done as follows:

1. All the exercises that are present in the training set of the Duolingo dataset were collected and then subdivided into multiple contiguous chunks of alternating teaching and testing data.The user will be "taught" using the teaching set and then evaluated using the testing set.
2. One pair of contiguous teaching and testing data subsets is used by the RL agent in a single episode.
3. Before the beginning of the episode, the user is be tested on the testing subset of the data and the score/accuracy of the user of the user is recorded. Next, the RL agent starts to schedule the exercises from the teaching subset for the user (using the ϵ greedy policy as mentioned before). At every step, the RL agent can either select one of the exercises from the teaching subset to present to the user or it can terminate the episode.
4. Once the episode terminates the user is evaluated again on the testing subset and the accuracy will be recorded.
5. By default the RL Agent is provided a reward of "-1" for every exercise that it schedules for the user. Once the episode terminates, the reward for the RL agent is computed as the difference difference between the final accuracy and the initial accuracy on the test subset of the data. This encourages the RL agent to schedule less number of exercises for the user while still ensuring that the knowledge gain of the user is not compromised.
6. This is repeated for each pair of teaching and testing subset of exercises in the dataset.
7. At every step (i.e. every time the RL agent selects an exercise and gets a reward) the linear regression model is updated using the gradient descent update rule for Q-learning.

4.3 T5 Training Procedure

Google's T5 pre-trained model that can be imported via HuggingFace was selected by us since it provides the most flexibility in it's retraining procedure and because of the plethora of documentation available for the same. Where the T5 is concerned, you can specifically train it for a dialogue generation task as well; which is what we want to do. This in turn would be connected to the RL agent which would be able to feed it the exercises chosen by the agent.

4.3.1 Dataset Modification

The dataset needs to be formatted in a specific format, which follows a two column structure. The first would be the prompt and the second would be the answer. This allows the model for a customized

dialogue generation task and also allows possibly allows for our Reinforcement Learning agent to feed in a prompt whenever necessary. In addition to this structure, we require to add a prefix into the prompt of the data, so as to make the T5 understand the task. In this case, the prefix we chose was "dialogue generation: ". All of this becomes more understandable when you see Figure 4 which provides a concrete example of the tuned dataset. After that, the data, was simply split into train, validation and test sets.

4.3.2 Preprocessing

While we did have a tuned dataset, we require a tokenizer to feed it into the model. For this purpose, HuggingFace's AutoTokenizer was used. Here, the maximum input length and maximum target length; which are essentially limits set on the generated sequences while training, were taken as 1024 and 64 respectively.

4.3.3 Hyperparameters

While we've already covered the input and output sequence lengths, a few ablations were required to reach the following hyperparameters. The batch size was taken to be 64, weight decay 0.01, and since the total dataset was quite huge, we trained it temporarily on three epochs.

Prompt	Answer
dialogue generation: Yo soy un niño.	Think you can convert this Spanish sentence [Yo soy un niño.] to English as well?
dialogue generation: Yo soy de México.	One more never hurt! Convert this Spanish sentence [Yo soy de México.] to English to seal the deal
dialogue generation: Mi nombre es Pedro.	Please convert this Spanish sentence [Mi nombre es Pedro.] to English
dialogue generation: Ella es una niña.	Please convert this Spanish sentence [Ella es una niña.] to English
dialogue generation: ¿Él es un niño?	One more never hurt! Convert this Spanish sentence [¿Él es un niño?] to English to seal the deal
dialogue generation: Yo necesito un taxi.	Please convert this Spanish sentence [Yo necesito un taxi.] to English
dialogue generation: ¿Dónde?	Think you can convert this Spanish sentence [¿Dónde?] to English as well?
dialogue generation: Yo tengo una reservación.	Think you can convert this Spanish sentence [Yo tengo una reservación.] to English as well?
dialogue generation: Yo estoy bien.	Please convert this Spanish sentence [Yo estoy bien.] to English
dialogue generation: Mi periódico	Think you can convert this Spanish sentence [Mi periódico] to English as well?

Figure 7: Tuned dataset for T5

5 Results and Analysis

5.1 Knowledge Tracing

For our Knowledge Tracing we achieve a validation scores 0.4246 (F1 Score) and 0.7732 (AUC-ROC) and a test scores of 0.4311 (F1 Score) , 0.77 (AUC-ROC) for the Duolingo Dataset.

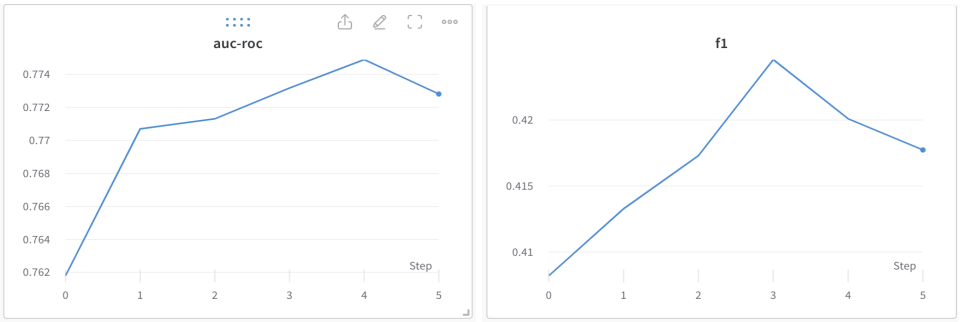


Figure 8: AUC and F1 scores of the knowledge tracing model on the validation set

The above-mentioned are results from the best performing model that we implemented. Apart from the implementation of the knowledge model mentioned above, the following are some of the experiments we tried:

1. Making use of multiple layers of multi-head attention instead of BiLSTM layers : This approach did not work as well as our current results possibly because of the difficulty in training multiple layers of multi-head attention based transformer blocks from scratch.
2. Making use of Attention to combine the final hidden states from the encoder into a single knowledge state vector (using a weighted average with the attention weights) : The results from the attention based approach were similar to the results obtained with max-pooling but we could not make use of attention in our final model since the computation of the attention requires the query vector from the decoder (i.e. the representation of the token for which we need to compute the prediction). This does not match with the requirements of the reinforcement learning agent (where we need a single representation of the knowledge state at any given time)
3. Making use of sum, mean pooling instead of max pooling to condense the final hidden state of the encoder into a single vector: Making use of max-pooling provided us with the best results possibly because a lot of the information in the input is contained locally and making use of max-pooling allows us to most effectively capture all the local pieces of information (as compared to mean pooling which borrows information equally from each time-step)

5.2 Reinforcement Learning Agent

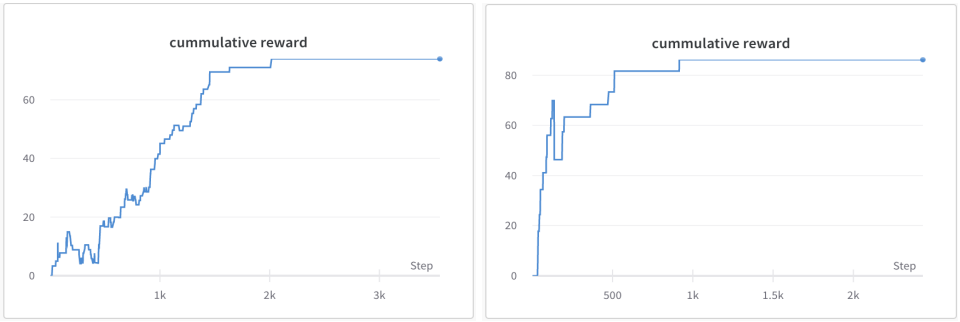


Figure 9: The cumulative score of our baseline RL model against our model

The baseline reinforcement learning agent requires approximately 1500 timesteps to reach saturation.

In contrast, our reinforcement learning module achieves saturation at a greater cumulative reward with merely 300 samples when the knowledge state is provided, and around 500 timesteps when the knowledge state is absent.

By surpassing the baseline approach in both cumulative reward and sample efficiency, our Q-learning based reinforcement learning agent demonstrates superior performance.

5.3 Analysis

5.3.1 Knowledge Tracing

Our knowledge tracing model performs slightly below the baseline models. This may be due to the fact that we exclude exercise format and userId features which do not occur in day-to-day conversations, but are included in the baseline models which are specific to Duolingo exercises.

Secondly, in order to input the entire state into our reinforcement learning agent we also compress the user's knowledge state into one single vector, unlike top-performing baselines which do not use RL agents.

5.3.2 Reinforcement Learning

A plausible explanation for the improved performance of our RL agent over random strategy could be that the agent comprehends the relationship between the knowledge state and the sentence-bert embeddings generated for each conversation. This understanding enables the RL agent to optimise its selections of conversation, focusing on the sentences that best enhance the knowledge state.

6 Conclusion and Future Work

Given that Reinforcement Learning performed better than random policy (saturating at around 300 episodes), we can conclude that it is possible to integrate with an online knowledge tracing model to better select exercises and improve user learning. This successful proof of concept for a reinforcement learning agent combined with knowledge tracing provides foundation for a multitude of future works. This agent could be fine tuned for more superior performance, and more naturally integrated into a second language acquisition system. Further, we believe that this technique could be used for a variety of tasks beyond just second language acquisition. Any online learning method which can track the known and unknown knowledge from the target knowledge task can benefit from this approach.

Our immediate future work goals are to deeply analyze the RL policy that has been learned by our agent. This will help us further tune and engineer our solution; and possibly better structure future datasets for similar tasks. We would also like to explore different reward strategies for the RL agent in-order to prioritize different exercise selection strategies (such as making sure that there is a balance between unknown and known concepts in the exercises that are selected by the agent). In addition, generalizing to any broad task with a structured dataset would be challenging. Using a generative model to generalize the RL agent to any task could greatly improve the learning experience that results from this method. A generative model that integrates with a knowledge tracing model and RL agent such as ours would be a great further application of our research.

References

[1] Jonathan Bassen, Bharathan Balaji, Michael Schaarschmidt, Candace Thille, Jay Painter, Dawn Zimmaro, Alex Games, Ethan Fast, and John C. Mitchell. How to train your learners: Reinforcement learning for the scheduling of online learning activities. In *CHI 2020*, 2020.

[2] Vito Bellini, Angelo Schiavone, Tommaso Di Noia, Azzurra Ragone, and Eugenio Di Sciascio. Computing recommendations via a knowledge graph-aware autoencoder. *arXiv preprint arXiv:1807.05006*, 2018.

[3] Albert T. Corbett and John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4:253–278, 2005.

[4] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*, 2018.

[5] Deigant et al. https://github.com/deigant1998/introtodeeplearning11785project, 01 2023.

[6] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018.

[7] Elise WM Hopman and Maryellen C MacDonald. Production practice during language learning improves comprehension. *Psychological science*, 29(6):961–971, 2018.

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[9] Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K Reddy. Deep reinforcement learning for sequence-to-sequence models. *IEEE transactions on neural networks and learning systems*, 31(7):2469–2489, 2019.

[10] Kostis. Deep learning for second language acquisition modeling (https://github.com/kostis-s-z/dl_4_slam), 09 2020.

[11] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[12] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*, 2017.

[13] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[14] Pierre Lison and Jörg Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. 2016.

[15] Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo. Graph-based knowledge tracing: modeling student proficiency using graph neural network. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 156–163, 2019.

[16] Nihal Nayak. Context based approach for second language acquisition (https://github.com/nihalnayak/slam18), 05 2022.

[17] Anton Osika, Susanna Nilsson, Andrii Sydorchuk, Faruk Sahin, and Anders Huss. Second language acquisition modeling: An ensemble approach. 2018.

[18] Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*, 2019.

[19] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[20] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. *Advances in neural information processing systems*, 28, 2015.

[21] Sherry Ruan, Liwei Jiang, Qianyao Xu, Zhiyuan Liu, Glenn M Davis, Emma Brunskill, and James A Landay. Englishbot: An ai-powered conversational system for second language learning. In *26th international conference on intelligent user interfaces*, pages 434–444, 2021.

[22] Kazuya Saito and Yuka Akiyama. Video-based interaction, negotiation for comprehensibility, and second language speech learning: A longitudinal study. *Language learning*, 67(1):43–74, 2017.

[23] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[25] Burr Settles, Chris Brust, Erin Gustafson, Masato Hagiwara, and Nitin Madnani. Second language acquisition modeling. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 56–65, 2018.

[26] Xiangyu Song, Jianxin Li, Qi Lei, Wei Zhao, Yunliang Chen, and Ajmal Mian. Bi-clkt: Bi-graph contrastive learning based knowledge tracing. *Knowledge-Based Systems*, 241:108274, 2022.

[27] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation, 2018.

[28] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning, 2019.

[29] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre M. Bayen, and Yi Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *ArXiv*, abs/2103.01955, 2021.

[30] Michael V. Yudelson, Kenneth R. Koedinger, and Geoffrey J. Gordon. Individualized bayesian knowledge tracing models. In H. Chad Lane, Kalina Yacef, Jack Mostow, and Philip Pavlik, editors, *Artificial Intelligence in Education*, pages 171–180, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[31] Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He. Relational graph neural network with hierarchical attention for knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9612–9619, Apr. 2020.

Appendix

A Baseline Models

The approach described in this project is quite novel and hence there wasn't any particular baseline that emulated what we intended to do on an end to end basis. For the purposes of reaching current performance metrics of the baselines models corresponding to the individual components (i.e. the knowledge tracing model and the RL based conversational agent), we implemented different sections of the proposed solution separately.

A.1 Knowledge Tracing

Most recently, Graph Neural Networks have been seen to perform better in knowledge tracing[15]; however, given our dataset, we do not have any graph oriented elements (on a superficial level). We're not inherently provided the skill-set required for each exercise which could be represented in a graph structure.

That said, we reviewed the winning solutions of the SLAM challenge conducted by Duolingo and saw that the first place solution was an ensemble between an RNN and a Gradient Boosted Decision Tree, which yielded them an AUC Score of 0.861 and an F1 score of 0.561. Hence, we chose this approach as our baseline for Knowledge Modelling and tried implementing various architectures based on that.

The following were the solutions that provided us with the most accurate results:

A.1.1 LSTM

We did manage to execute an LSTM architecture which was inspired by the work of [10]. They used GloVe [19] for creating the embedding of the tokens (words). In order to predict the label of a particular token in an exercise, they took into account the previous 50 tokens from the exercises that were solved by the user as context. This data was fed into a Simple LSTM to predict the labels for each token. The ablation details of the same have been provided below in section 5.

A.1.2 Augmented Logistic Regression

We also found an interesting approach by [16] which made use of the baseline Logistic Regression provided by Duolingo, but augmented it by taking into account the encoded featurs of the token preceding the current token(that has to be predicted). It was surprising that just taking one timestep's worth of context without any loss of information provided such a significant boost. The results and ablations of this implementation can be seen in section 5.

A.1.3 RL based Conversational Agent for language learning

We propose using a Reinforcement Learning Agent that, based on the knowledge tracing results of the user, will suggest the next exercises to learn for that particular user; thus making the whole learning experience more personalized. As far as the work is concerned we understand that it is very essential to have a sample-efficient policy optimization to ensure the model provides reasonable performance within small number of user inputs. In the case of language learning this is even more important given the fact that the user's learned state will continuously change on interacting with new learning modules. And hence, for a given state we will have very few data samples, at max the number of users who interact with the model.

As per our best knowledge there is no work that is using Reinforcement learning for language learning task on this particular dataset. Bassen et. al. [1] use Reinforcement learning scheduler for learning linear algebra.

Since we don't have an existing baseline for the RL based exercise selection, we have implemented a random policy agent; which out of the all the given exercises in the Duolingo dataset - randomly divides it into multiple teaching and testing subset pairs and then tests that policy against a simulated BKT based learner. The results for that can be seen in the next section.

Here are the links of a selected subset of the ablations that we've run for this project (along with a short description for the same):

A.2 LSTM Ablations:

A.2.1 200 epochs

This shows the ablation for LSTM that ran for 200 epochs and provided us with really good insights going forward: https://wandb.ai/idl-s23/DL_4_SLAM-starter_code/runs/d8lu5p8p?workspace=user-vinayn

A.2.2 50 epochs

This shows the ablation for LSTM that ran for 50 epochs which provided us with the best results by far, esepcially where the F1 score was concerned: https://wandb.ai/idl-s23/BaseLine%20Ablations/runs/m5l9b71x?workspace=user-vinayn

A.3 Augmented Logistic Regression

A.3.1 60 epochs

This shows the ablation for the augmented Logistic Regression Model that ran for 60 epochs. This was the one that gave us best results for AUC Score: https://wandb.ai/idl-s23/BaseLine%20Ablations/runs/cehsz55d?workspace=user-vinayn

A.3.2 30 epochs

This shows the ablation for the augmented Logistic Regression Model that ran for 30 epochs. Although we ran it with different hyperparameters, the performance was similar to the one with 60 epochs: https://wandb.ai/idl-s23/BaseLine%20Ablations/runs/o6yu27j3?workspace=user-vinayn

A.4 Conversational Agent

A.4.1 3500 subsets

This shows the ablation for the BKT (Bayesian Knowledge Tracing) Simulated Learner. The training subset was of size 50 and that of testing was of size 20. Computed on the basis of all the training subsets played to the user. The X-axis is the total number of training subsets played to the user: https://wandb.ai/idl-s23/BaseLine%20Ablations/runs/o6yu27j3?workspace=user-vinayn

A.5 Baseline Implementations

As mentioned before, we have two major subsections of the proposed solution:

A.5.1 Knowledge Tracing

LSTM The LSTM baseline we implemented was based on the model and code referenced in Julia et. al. [10]. Their implementation involved taking GloVe embeddings of the tokens and loading it in the form of chunks.

In this implementation, the LSTM takes into account 50 previous timesteps as context of that word and predicts the output label at the last timestamp. However, the entire dataset's embeddings couldn't be fed due to memory constraints and hence we ended up splitting the data into chunks of data; for both training and testing purposes.

Augmented Logistic Regression As opposed to the LSTM method mentioned above, the Logistic Regression method inspired by Nihal V. Nayak et. al. [16] doesn't convert the words into embeddings. Instead, it hard-codes the one-hot encodings of each word while feeding it into the model.

That said, it also takes into account the features of it's previous timestep while processing. The only difference between their model and the baseline model is the fact that they take into account the previous state of the input which resulted in a significant boost of performance as compared to the original baseline of the competition.

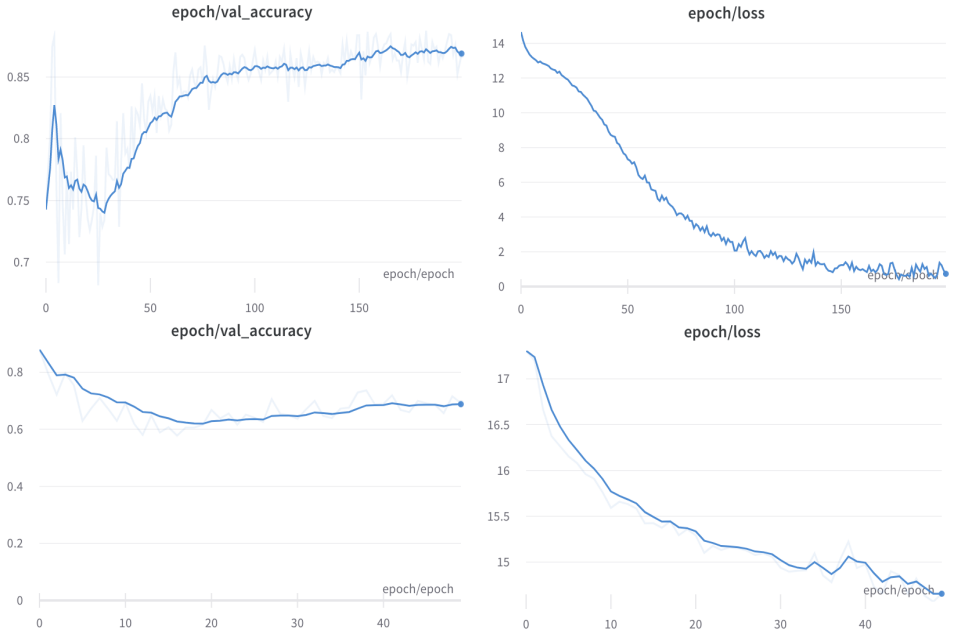


Figure 10: Training curves for LSTM network - The top two were run for 200 epochs while the bottom two were those ablations that were run for 50

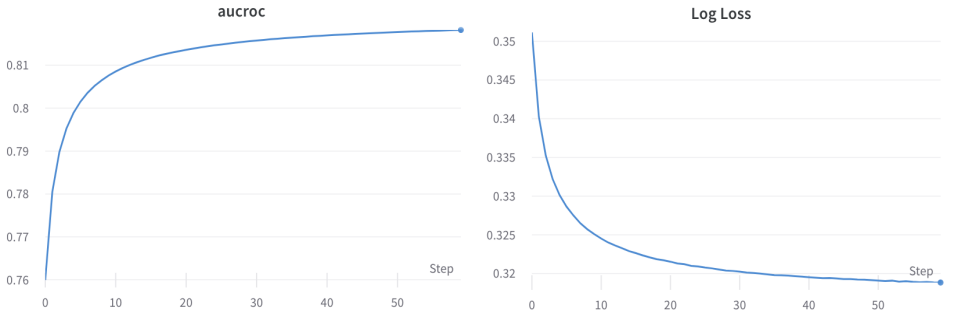


Figure 11: Training Curves for the Augmented Logistic Regression Model

Light Gradient Boosted Decision Tree Current state of the art knowledge tracing models ensemble multiple classifiers to maximize knowledge tracing performance. The most effective ensemble method integrated a Light Gradient Boosted Decision Tree as an additional knowledge tracing model. We implemented a Decision Tree following the methods and architecture implemented by Osika et al., which achieved the best performance in the 2018 Duolingo SLAM competition.

The Decision tree implemented by Osika et al. required additional feature engineering beyond the baseline features provided by duolingo in the dataset. Further engineering allows for extracting additional features including the number of times the user has previously seen the token, where the token is in the sequence, and how long it has been since the user has last seen the specific token. These additional features reportedly allowed for significantly higher performance by the developers in Osika et al. Surprisingly, our replications of this model actually performed worse with those additionally engineered features than without. Lack of specificity in the referenced paper prevented us from being able to determine what differences in our feature representation resulted in such a large gap in performance.

The architecture reported by Osika et al. contained 2400 leaves, 3203 estimators, a learning rate of 0.005, and 40% feature fraction use for splitting. We used this same architecture, but got better performance without the additional features. The performance is reported below.

Table 1: Results from Baseline Ablations

Model	Epochs	AUC-ROC	F1 Score
LSTM	40	0.751695	0.389982
Logistic Regression	60	0.818203	0.354804
LGBM	N/A	0.6127	0.35785
LGBM (Additionally Engineered Features)	N/A	0.539419	0.151575

A.5.2 Conversational Agent

Results of a random policy for exercise selection In-order to establish a baseline for the RL based exercise selection agent, we created a random policy and evaluated it against a BKT based Simulated Learner [5]. The following graph shows the cumulative rewards of the random policy :

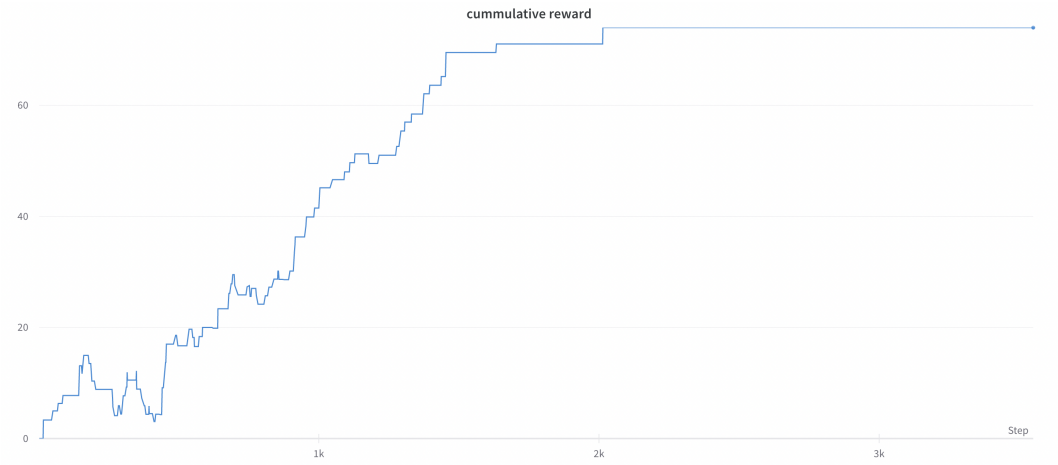


Figure 12: Cumulative reward for a random policy

In-order to perform the above test, we divided the dataset into pairs of teaching and testing subsets containing 50 and 20 exercises each respectively. From the above cumulative graph curve, we can see that it takes about 2000 teaching batches/data subsets for the BKT learner to achieve mastery for all the content that is present in the Duolingo dataset.

Sample efficient reinforcement learning We ran ablations of sample-efficient RL policy optimizations on a toy-playground of OpenAI-Gym, with the playground "Pendulum-v1". We use actor critique method for policy update. We compare the actor-critique method sample-efficiency of Vanilla policy gradient, PPO, TRPO, TD3, behavior cloning and DDPG. We observe that PPO, TD3 and behaviour cloning provide much better sample efficiency over Vanilla policy gradient.

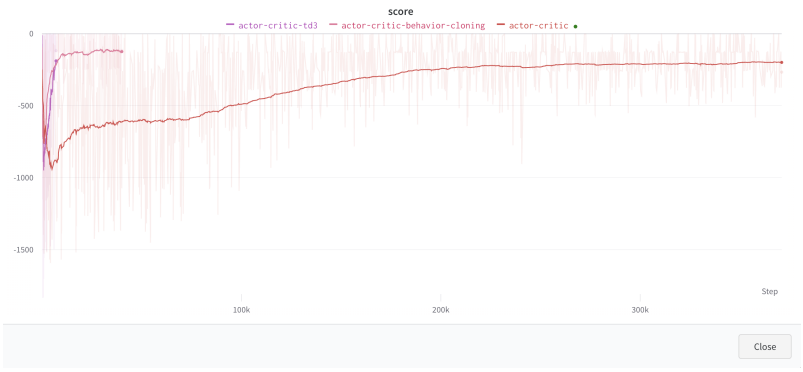


Figure 13: Reward vs No. of steps for different policy optimization techniques(Actor critic-TD3, Actor-Critic DDPG, Vanilla Actor-Critic)