Executive Summary of the Thesis

A study of possible improvements in Knowledge Tracing with Natural Language Processing and self-attention

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Simone Sartoni

Advisor: Prof. Paolo Cremonesi

Co-advisor: Luca Benedetto, PhD

Academic year: 2021-2022

1. Introduction

In the educational domain, Knowledge Tracing (KT) is the task of modelling the knowledge of a student over time, aiming at understanding its ability levels in different subjects. Inferring unobservable traits from their observed performances on assessments enables us to keep track of student level, customizing the learning process and enriching the experience of online education. For example, KT can be useful in online assessments to recommend particular questions, directly targeting students' weaknesses. In our work we focus and compare the models on a specific sub-task of KT: answer correctness prediction. Given a student S, we denote the act of submitting answer r_t to exercise question e_t at time t as an interaction $I_t = (e_t; r_t)$. Answer correctness prediction can be mathematically described as the the task of predicting value r_T from past interactions (e_0, r_0), (e_1, r_1), ... $(e_{T-1},\ r_{T-1})$ and target exercise e_T. To make better predictions, some KT models assume having at least an ability (or "skill") s_t associated with each question q_t, providing context information (such as the main topic of the question or the required knowledge). Knowing the skill enables an easier representation of knowledge, but at the same time, the performance of the models based on this assumption can be limited by its availability.

2. Related works

Different architectures have been proposed to model KT and predict answer correctness. Initially, probabilistic approaches (such as Bayesian Knowledge Tracing) tried modelling directly the human behaviour of knowing a skill, acquiring the knowledge about it after an interaction, guessing or slipping it (answering wrongly despite knowing the skill).

2.1. Deep Knowledge Tracing

In 2015, Deep Knowledge Tracing (DKT) [3] showed the capabilities of Recurrent Neural Networks (RNN) in modelling this task, outperforming BKT on most of the datasets. DKT can mimic a function considering latent concepts, the difficulty of each exercise, the prior distributions of student knowledge and its evolution over time. Since DKT needs a large amount of data, it is well suited only for online education. In our work, we consider as the first baseline a powerful variant of DKT, implemented using Long Short Term Memory units instead of recurrent ones.

2.2. Attention-based models

Recently, some works have developed attention-based models for KT because attention has outperformed deep neural networks in many other tasks. In particular, Self Attentive Knowledge Tracing (SAKT) is the first model to apply self-attention to KT. It improves the Area Under the Curve metric and enables the parallelization of the computation, making the model orders of magnitude faster than DKT. Instead, Separated Self-AttentIve Neural Knowledge Tracing (SAINT) [1] has introduced the Transformer model to KT, which is a more complex encoder-decoder architecture with self-attention layers as basic blocks. Furthermore, SAINT has been extended to use temporal features in the so-called SAINT+ [4] architecture. SAINT+ is the second baseline we compare our models with.

2.3. Natural Language Processing for KT

A recent direction to improve KT is to generate additional information about the questions from their texts. For example, transforming raw text into useful fixed-length vectors and developing KT models working on generic vectors as input, can enable to use together information from heterogeneous sources, such as text, skill and temporal features. Several works have investigated the use of a single Natural Language Processing technique to improve an LSTM or a self-attention architecture. Exercise-Enhanced Recurrent Neural Network (EERNN) [5] produces an embedding for each word with word2vec and uses a bidirectional LSTM network to learn relations between words and the last hidden state as the textual embedding for the exercise. Then EERNN adopts an LSTM network or an attention mechanism to produce predictions. Lastly, Relation-Aware Self-Attention for Knowledge Tracing (RKT) [2] is an extension of SAKT made by the same authors, preprocessing similarity coefficients between exercises, based on: i) text similarity, i) time between the interactions and iii) the performance of users. Those additional coefficients are summed to the outputs of the SAKT self-attention layer. Other works (such as EKT and EHFKT) have provided other solutions, but they are not relevant to our work.

3. Proposed models

Our work starts from the consideration that self-attention and Natural Language Processing are still underexplored topics to improve KT, so we decide to focus on them. In our work, we follow two directions to improve KT models. First, we propose Prediction Oriented Self-attentive knowledge Tracing (POST), a self-attention based model composed of three components: an encoder and two decoders. Then we examine six Natural Language Processing methods (i.e. CountVectorizer, word2vec, doc2vec, DistilBERT, Sentence Transformer and BERTopic) to produce embeddings from the exercises' texts and develop "NLP-enhanced" versions of DKT and POST, able to use as input the textual embeddings. Furthermore, we propose two approaches to create an "hybrid" NLP-enhanced DKT model able to use at the same time different NLP methods.

3.1. Generate textual exercise embeddings

In our work, we identify six NLP methods to create exercise embeddings from exercise text: CountVectorizer, word2vec, doc2vec, DistilBERT, Sentence Transformer and BERTopic.

3.1.1 CountVectorizer

CountVectorizer collects all the n_w words, associating each of them with a unique index. Then, it represents a text as a vector with dimension n_w, where the value at position i is the count of times the word associated with index i appears in the text. This method can lead to a high number of words. We can reduce it with three techniques: i) using a sparse implementation of the matrix used to represent all the texts, ii) lemmatizing words and removing stopwords and iii) removing words appearing less than min_df times, more than max_df times, or maintaining only the most frequent $max\_features$ words.

3.1.2 Word2vec and DistilBERT with average operator

Word2vec and DistilBERT are NLP models to create a unique vector with a fixed length for each word appearing in a collection of texts. We can use them to produce a set of word embeddings for each text and then apply the average function over the words dimension, generating a single exercise embedding with the same length. Exercise embeddings need to be normalized to be useful for KT. Word2vec model consists of a neural network with one layer. Instead, DistilBERT is obtained "distilling" BERT model, a multi-layered Bidirectional Transformer pretrained on Masked Language Modeling and Next Sentence Prediction tasks. Word2vec requires removing HTML tags, lemmatising words and removing stopwords from texts, while BERT can work directly with text without HTML tags.

3.1.3 Doc2vec, Sentence Transformer and BERTopic

Doc2vec is a good alternative to averaging word2vec embeddings, enabling to learn at the same document and word embeddings. We can directly use document embeddings as exercise embedding for KT. We denote as "Sentence Transformer", the Siamese network built on top of *all-mpnet-base-v2* Transformer. It creates exercise embeddings as outputs of the Pooling and Normalization layers applied to word embeddings generated with *all-mpnet-base-v2* Transformer architecture. We use the implementation of this model from SentenceTransformers[1] library, a python framework to create state-of-the-art sentence, text and image embeddings. Differently from previous methods, BERTopic does not embed text. It can automatically learn the topics of a collection of texts and compute the most probable one or a probability vector, associating with each one a probability.

3.2. NLP-enhanced DKT

To check the utility of the six NLP methods, we develop NLP-enhanced DKT, an LSTM network taking as inputs the exercise embeddings. The idea is to reproduce DKT with float vectors as inputs. We embed interaction (e_t, r_t) at time step t as in EERNN:

$$\tilde{X}_t = \begin{cases} [X_t \oplus \mathbf{0}] & if\ r_t^i = 1 \\ [\mathbf{0} \oplus X_t] & if\ r_t^i = 0 \end{cases} \quad (1)$$

where $\oplus$ is the concatenation operator and X_t is exercise embedding associated to e_t, generated by a NLP method. The sequence of embedded interactions $(\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_t)$ is now used as input to the LSTM network. LSTM outputs $h_t \in R^{d_h}$ are passed to a Dense layer with sigmoid activation function, producing outputs $\tilde{X}_o \in R^{2 \cdot d_{embeddings}}$. First $d_{embeddings}$ elements of $\tilde{X}_o$ can be seen as positive knowledge representation, while the second half is negative one. Now our model takes the "target" exercise embeddings X_{t+1} and compute element-wise multiplication to both the positive and negative half of $\tilde{X}_o$. In the end, a Dense layer is applied to predict $\tilde{r}_{t+1}$ correctness probability.

[1]https://www.sbert.net/docs/pretrained_models.html

3.3. Prediction Oriented Self-attentive knowledge Tracing

We propose two new models based on self-attention: Prediction Oriented Self-attentive knowledge Tracing with Multiplication (POST-M) and Prediction Oriented Self-attentive knowledge Tracing (POST), both composed of a past exercise content encoder, a past performance decoder and a prediction oriented module. We underline that all the encoders and decoders use Masked Multi-Head Attention to avoid invalid attending, as in SAINT+. We use three matrices to create embeddings with same dimension for each exercise id e_t, skill id s_t and answer correctness r_t. Past exercise content encoder receives as query, keys and values the sum of the embeddings related to exercise, skill and a positional encoding: $E_i^e = e_i^e + s_i^e + p_i$, but only for past time steps from $T = 0$ to $T = t$, padded with an initial start token. The past performance decoder use as queries the sum of embeddings related to answer correctness, elapsed time and positional encoding: $R_i^e = r_i^e + et_i^e + p_i$, from time step $T = 0$ to $T = t$, with an initial start token. Instead, keys and values are the outputs from the past exercise content encoder. In the end, prediction oriented module is responsible of combining two vectors: the performance decoder outputs y_{dec} and the target exercise embedding $E_{t+1}^e = e_{t+1}^e + s_{t+1}^e$ (without positional encoding) to predict the correctness r_{t+1}. POST-M computes Hadarmard product (also called element-wise product) of the inputs, while POST uses a decoder, taking as query the target embedding E_{t+1}^e and, as key-value pairs, the outputs of past performance decoder. In the end, both POST-M and POST have a linear layer to produce predictions.

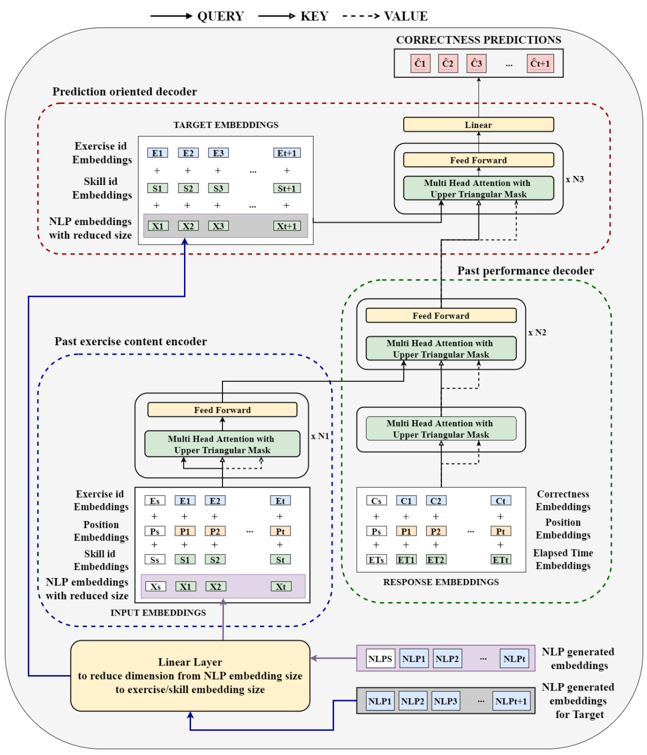

Figure 1: The architecture of NLP-enhanced POST model.

3.4. NLP-enhanced POST

Then we develop NLP-enhanced POST: an extension of POST able to use as inputs the textual exercise embeddings (§ 3.1). NLP-POST applies to each textual embedding a trainable Linear layer, computing $\tilde{X}$:

$$\tilde{X}(X) = f(W^T X + b) \qquad (2)$$

where $\tilde{X} \in R^{n_{dim}}$ and n_{dim} is the embedding size of the POST model. Then it sums $\tilde{X}$ to the input embedding of POST. In Figure 1 we report the architecture of NLP-POST.

3.5. Hybrid approaches

In the end, we implement two approaches to create a hybrid NLP-DKT, able to use at the same time textual embeddings generated by multiple NLP methods. The hybrid approaches we present consist of using two or more NLP-DKT architectures with different NLP methods and combining their outputs into a single one. For example, we can:

- compute the final prediction as a weighted sum of the predictions of parallel models;
- compute the final prediction by applying a Dense layer with one output to the concatenation of the outputs of the multiply blocks of the parallel models.

4. Experimental setups

In our work we use four datasets: ASSISTments[2] 2009 (AM09) and 2012 (AM12), Cloud Academy[3] (CA) and Peking Online Judge[4] (POJ). Each dataset consists of two documents, containing respectively interactions and textual information. The following operations are applied to the datasets:

- clean the texts, removing HTML tags, links, digits, punctuation and special characters. The produced output, denoted as *plain text*, is used as input to DistilBERT, Sentence Transformer and BERTopic. Furthermore, remove stopwords, lemmatize words and tokenize them, creating *clean sentence*, used as input to CountVectorizer, word2vec and doc2vec.
- Remove interactions without an available text or whose *clean sentence* has only one word.
- Remove duplicated interactions.
- Group interactions regarding the same user into sequences. Chunk them into subsequences of maximum length $N = 500$ (or $N = 100$ for self-attention based models). Remove sequences with a single interaction and pad to length N. Number of users and sequences are shown in Table 1

N=500	AM09	AM12	CA	POJ
# **users**	3,836	45,975	17,021	10,557
# **sequences**	3,346	45,375	17,378	2,102

Table 1: Number of users and subsequences for each dataset.

After processing, we perform a training/validation/test split on sequences of 60%/20%/20%. Batch size varies according to the dataset and the model, due to memory limitations. Models are trained minimizing the Binary Cross Entropy loss and evaluated with

[2]`https://sites.google.com/site/assistmentsdata/`
[3]https://cloudacademy.com/
[4]http://poj.org/

Binary Accuracy (ACC) and Area Under the ROC Curve (AUC) metrics. Dropout rates are fixed to 0.3 for LSTM models and 0.2 for self-attention ones. Each model is trained and evaluated using three learning rates ($lr = 1e{-3}$, $lr = 1e{-4}$ and $lr = 1e{-5}$), but only the best result is reported.

5. Results

In our work, for each dataset, we consider four baselines: the majority prediction model, DKT using exercise ids, DKT using skill ids (whenever available) and SAINT+, and we compare their ACC and AUC with POST-M and POST. Then, we evaluate NLP-DKT and NLP-POST with the six NLP methods. In the end, for each dataset, we consider the best two (or three) NLP methods and evaluate the two approaches to create hybrid models for NLP-DKT using those NLP methods. We evaluated more than seventeen models per dataset, so we present only a summary view of them. This view contains the results of the best performing:

- baseline;
- model not using textual embeddings;
- NLP-enhanced model and respective NLP method;
- hybrid approach for NLP-DKT.

In particular, in Tables 2, 3, 4 and 5 we show the results of these models respectively in AM09, AM12, CA and POJ datasets.
As in literature, we evaluate KT models by comparing their AUC and binary accuracy.

Best	**model**	**metrics**	
		ACC	**AUC**
baseline	DKT skills	0.711	0.721
without NLP	POST	0.707	0.736
NLP-enhanced	NLP-DKT CountVectorizer	0.735	0.778
hybrid approach	sum of predictions	**0.742**	**0.784**

Table 2: Results of the best performing models on ASSISTments 2009 dataset. Hybrid approach uses CountVectorizer, DistilBERT and word2vec.

Best	**model**	**metrics**	
		ACC	**AUC**
baseline	SAINT+	0.707	0.736
without NLP	POST	0.754	0.790
NLP-enhanced	NLP-POST Sentence Transformer	**0.757**	**0.793**
hybrid approach	multiply blocks	0.745	0.775

Table 3: Best performing models results on ASSISTments 2012 dataset. Hybrid approach uses DistilBERT, Sentence Transformer and word2vec.

best	**model**	**metrics**	
		ACC	**AUC**
baseline	SAINT+	0.692	0.759
without NLP	POST	0.699	0.769
NLP-enhanced	NLP-POST CountVectorizer	**0.703**	**0.771**
hybrid approach	sum of predictions	0.666	0.699

Table 4: Results of the best performing models on Cloud Academy dataset. Hybrid approach uses CountVectorizer, DistilBERT and doc2vec.

best	**model**	**metrics**	
		ACC	**AUC**
baseline	SAINT+	0.618	0.661
without NLP	POST	0.634	0.688
NLP-enhanced	NLP-POST word2vec	0.644	**0.703**
hybrid approach	multiply blocks	**0.653**	0.696

Table 5: Results of the best performing models on POJ dataset. Hybrid approach uses DistilBERT, BERTopic and word2vec.

6. Conclusions

In our work, we studied multiple directions to improve the answer correctness prediction task. First of all, our results showed that SAINT+ works better than DKT on large datasets (such as AM12 and CA), while it is the opposite on small datasets with available skills (AM19).

6.1. POST

Initially, we proposed Prediction Oriented Self-Attentive knowledge Tracing (POST), a novel self-attention model for KT. In SAINT+, the encoder has two objectives: learning what is relevant in the embeddings and comparing past ones with the target one. Instead, in POST, the two objectives are given respectively to the past exercise content encoder and the prediction oriented decoder.
The summarized results show that POST has higher AUC than all the baselines on each dataset, proving that POST models the answer correctness prediction task better than SAINT+. In addition, POST has the advantage to work optimally even with a small number of training samples (such as in AM09).

6.2. NLP-enhancing and hybrid approaches

Then, we examined six NLP methods to produce textual embeddings and developed “NLP-enhanced" versions of DKT and POST. The significant improvements in their results confirm the utility of using texts to improve KT. However, the results depend too much on the dataset to provide a general ranking of the NLP methods.
In particular, NLP-enhancing shows the best utility on small datasets, furnishing additional relevant content information. Instead, on large datasets (AM12 and CA), NLP-POST still improves the results, but it has identical performance with all the NLP methods, meaning that the textual exercise embeddings are mainly ignored. Some possible reasons are:

- POST can learn the textual relations from the large number of interactions.
- the linear layer limits the use of textual embeddings.
- summing textual embeddings to skill and item ones reduces results.

We suggest studying new approaches to add textual embeddings to self-attention models, such as creating a new encoder responsible only for textual content or replacing the linear layer with new components.
Lastly, we presented the possibility of creating “hybrid” models, using, at the same time, exercise embeddings produced by multiple NLP methods. Hybrid NLP-DKT has improved the results of NLP-DKT on each dataset, proving their utility. In particular, it provided the highest AUC on AM09 dataset, improving by 8.73% the AUC of DKT.
To summarize the results, our proposed models increased the AUC metric of previous models by 8.7%, 7.7%, 1.6% and 6.4% respectively on AM09, AM12, CA and POJ datasets; while the binary accuracy increased by 4.2%, 7.1%, 1.6%, and 5.7% respectively.

References

[1] Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungsoo Kim, Yeongmin Cha, Dongmin Shin, Chan Bae, and Jaewe Heo. Towards an appropriate query, key, and value computation for knowledge tracing. *CoRR*, abs/2002.07033, 2020.

[2] Shalini Pandey and Jaideep Srivastava. RKT : Relation-aware self-attention for knowledge tracing. *CoRR*, abs/2008.12736, 2020.

[3] Chris Piech, Jonathan Spencer, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J. Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. *CoRR*, abs/1506.05908, 2015.

[4] Dongmin Shin, Yugeun Shim, Hangyeol Yu, Seewoo Lee, Byungsoo Kim, and Youngduck Choi. Saint+: Integrating temporal features for ednet correctness prediction. In *LAK21: 11th International Learning Analytics and Knowledge Conference*, LAK21, page 490–496, New York, NY, USA, 2021. Association for Computing Machinery.

[5] Yu Su, Qingwen Liu, Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Chris Ding, Si Wei, and Guoping Hu. Exercise-enhanced sequential modeling for student performance prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.