Personalized Online Self-Learning

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Shalini Pandey

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Prof. George Karypis and Prof. Jaideep Srivastava

May, 2021

Acknowledgements

My PhD journey started in 2016 and looking back now, the whole journey has made me a stronger person and taught a lot of life lessons. This journey would not have been possible without help of my friends, family and mentors.

First and foremost, I would like to thank my advisors, Prof. Jaideep Srivastava and Prof. George Karypis. When I got admission at University of Minnesota, I first talked to Prof. George Karypis and he has been a great source of inspiration since then. His hard-work and brilliance is unparallel and I had a lot of opportunities to learn research skills from him. I admire his style, courage and hard-work and aspire to learn those qualities one day.

I do not have enough words to thank Prof. Jaideep Srivastava. He always showed confidence in me and kept encouraging me to achieve more. He genuinely cares for everyone and brings the best out of them. In addition to providing sufficient guidance on academic work, he showed me the value of being compassionate and treating everyone with respect. I wish to ingrain these traits in my future endeavors.

I would also like to thank my committee members, Professors Maria Gini, and Panayiota Kendeou for being accessible when requested to serve at my, thesis proposal, and thesis defense examination. Additionally, thanks to Professors Haiyi Zhu, and Ned Mohan for serving in my preliminary oral exam committee. They are all experts in their fields and my thesis would not have been in this position without their positive feedbacks and discussions.

I also learnt a lot from my other mentors during this PhD, particularly, Swayambhoo, Vineeth; Payal, Subhojit, Xia, Saurabh; Mas-ud and Martin. Through various internships, they taught me to apply the skills gained during PhD in industry.

Last but not the least, I would dearly want to thank all the current and past members

of GK lab and DMR lab with whom I have spent memorable times. They have been very supportive and discussions with them expanded my knowledge and provided invaluable feedback on my work. I would like to thank my maternal uncle and aunt who motivated me for higher studies and took care of me all these years. My teachers throughout my life had encouraged me and gave me the confidence to aim higher. I would also like to thank my friends and their respective whatsapp groups which kept me in touch of life outside grad school.

Dedication

Dedicated to my parents for their support and numerous prayers and blessings.

Abstract

With the growth of the internet, online learning platforms such as edX, Coursera, and Udacity have emerged. The Massive Open Online Courses (MOOCs) provided by these learning platforms are changing the landscape of education. The advantage of MOOCs is that they make courses available at a nominal price to students all across the globe. With the ability to reach a large number of learners around the world, MOOCs have made a positive impact on education. In addition, professional learners take these courses with the goal of achieving professional and career growth. This increases the audience size of the learning platform. Recent studies have shown that MOOCs have emerged as a disruptive technology with the potential of changing the shape of existing educational setting [1].

Despite the convenient settings provided by MOOCs, drop out rates on the learning platforms remain elevated. Some learners who drop out report lack of support by these platforms as a major reason for their disengagement. A factor contributing to this lack of personal guidance is that the online learning platforms follow *one-size-fits-all* and are not customized for different individuals. Currently, in most of the online education settings student has to determine everything, from what courses to pick to what questions to solve. Instead, an ideal learning system must scaffold the learning process—from initial modeling and coaching-oriented feedback to a gradual release of responsibility to students. Without sustained student input and feedback, their talents, creativity, and efficacy can be overlooked or negated. To tackle this problem, we need to develop systems that support self- learning. *Personalized self-learning* is defined as a teaching and learning process that assists learner based on the strengths, needs and interests of individual learners, while enhancing the self-learning experience. Massive data generated by online learning platforms has made research in this direction possible. Machine learning and data mining communities are focusing on application of AI in MOOC education research.

The first step leading to the development of personalized systems is to identify the needs of individual learners. In an online education systems, it is important that we determine the strengths and weaknesses of learners before customizing the platform

to their condition. A system to assess learner's knowledge can also help in proving a justification to learner regarding what they need to focus or what learning trajectory to follow.

Second, we personalize the recommendation of forums to improve the experience of students on MOOCs. The discussion forums have become an open source venue for sharing knowledge which generate auxiliary source of learning. For students taking the online courses, these auxiliary material can help interested student have a constructive discussion with their peers. However, it is difficult for them to browse through the enormous amount of forums to find the relevant thread of their interest.

Lastly, we aid self-learners who join the online learning system for developing a specific skill (such as machine learning) or learning a particular concept. Specifically, we provide them the pre-requisite concepts to master before focussing on their goal concept. We believe that this information can help the learners pick the concepts and videos to watch more intelligently. In addition, we also recommend next videos for learners to watch based on their interaction behavior in the past. For this we develop a novel representation learning technique that leverages the rich information about their textual content and structural relations between entities.

In summary, this thesis contributes towards development of a personalized MOOC platforms, specifically providing the following application 1)Knowledge Assessment to determine the strengths and weaknesses of students, 2) Forum recommendation to recommend relevant forums to students, 3) Concept Pre-requisite Prediction to predict pre-requisite relations between different knowledge concepts, and 4) Learning Path recommendation to recommend the sequence of videos a student needs to pick to achieve their goals.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the growth of the internet, online learning platforms such as edX, Coursera, and Udacity have emerged. The Massive Open Online Courses (MOOCs) provided by these learning platforms have changed the educational landscape across the globe. The advantage of MOOCs is that they make courses available practically free to learners from all over the world, thus breaking traditional barriers of time, place, and space. Learners enroll in these courses with various goals, including personal, professional, and career growth. However, these platforms still face various issues, which need to be addressed so that their full potential can be leveraged. *First*, MOOCs still provide a *one size fits all* system. From learning content to taking tests, MOOCs today largely resemble classroom teaching where learners fit within pre-determined parameters that leave little room for individualization [1]. *Second*, communication and information diffusion in MOOCs is limited, leaving learners to connect through other means and social networks [2], and thus the emergent learner behaviors are difficult to observe. *Third*, the overwhelming number of courses provided by the online learning platforms results in a severe overload of information for MOOC learners, who spend significant time surfing through the internet to find the course that suits them best [3], often leading to confusion, non-productivity and frustration. For these reasons, effective learning through MOOCs requires a different approach than currently available [4].

Today, it is well-recognized that effective learning through MOOCs requires different pedagogies than those used in classroom setting [4]. In MOOCs, students generate rich learning behavior data through interactions with the system. These interaction data has

been released for AI research by multiple MOOC platforms (e.g. XuetangX, Coursera and EdX) [5]. We propose to leverage this data for developing a personalized learning system. As shown in Figure 1.2, we envision this system as a personal guide for a learner. This system will assess student knowledge and provide them summary of their strengths and weaknesses, help them navigate through the course, recommending study material relevant to their interest and even, recommending a learning trajectory to acquire a specific skill. Developing such a personalized learning system raises several challenges in terms of design adopted for integrating various components, as well as developing new machine learning models to build each individual component. This research transforms MOOCs into testbeds for advancing educational research, and ultimately, improving learning.

1.1 Thesis statement

This thesis aims at developing a Personalized Online Self-learning System as a personal guide to a learner to improve learner's experience and retention.

To that end, we contribute several machine-learning models to various applications important for Education Data Mining (EDM) community. Particularly, we focus on learner knowledge assessment, interest prediction,and developing a guide to help learners plan their next activity.

1.2 Thesis Outline and Original Contribution

We start with providing the necessary background and overall motivation of this dissertation in Chapter 2. The remaining outline and major contributions of this dissertation in graph embedding literature is shown in Figure 1.1 and further discussed below in the given order,

1.2.1 Chapter 3: Vision and Design of POSLS

In this thesis we first propose our vision of Personalized Online Self Learning System (POSLS). We build a system that can be used by a user who wants to acquire some skills/knowledge on his own through the online media. There are many reasons

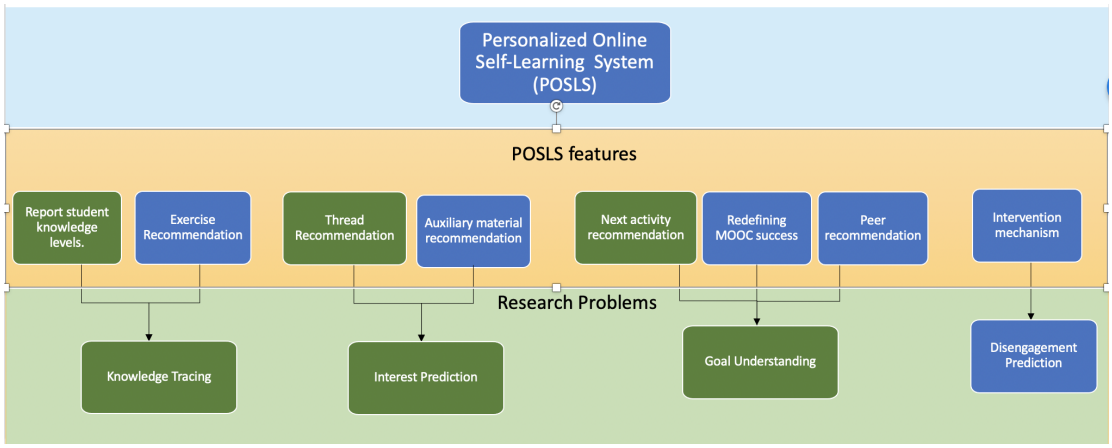


Figure 1.1: Above Figure shows the overall contributions of this dissertation towards POSLS. Green box represent the original thesis contributions.

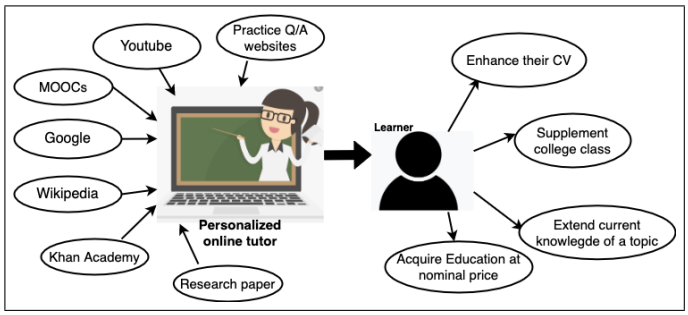


Figure 1.2: A self-learning tutor

which motivate a person to take up some online course such as, enhancing their CV, supplementing their college education, extend their knowledge of a topic or acquiring knowledge from an institution which they could not afford due to geographical or financial cause. The growth of internet has led to spawning of many online course provided by high class institution. However, with more sources of knowledge available, new challenges come in the scenario. First, it leads to an information overload. Second, most of these online learning systems are standard rather than personalized. Since learners on the online learning system belong to different backgrounds it is important for the system to customize based on individual needs. To address these issues, we propose **P**ersonalized **O**nline **S**elf **L**earning **S**ystem (POSLS) that acts as a personalized tutor

to guide a learner. This tutor gathers information from various sources and provide the individual learner a set path of learning activities suited by his/her learning preference. Figure 1.2 shows a tutor who utilizes information from all the sources and provide a student the relevant study material. Further details of design and architecture is present in the chapter 3

1.2.2 Chapter 4: Student Knowledge Modeling

The problem of modeling student knowledge is popularly called knowledge tracing, where we model each student's mastery of knowledge concepts (KCs) as (s)he engages with a sequence of learning activities. Each student's knowledge is modeled by estimating the performance of the student on the learning activities. The KT task can be formalized as a supervised sequence learning task - given student's past exercise interactions $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t)$, predict some aspect of his/her next interaction $\mathbf{x}_{t+1}$. On the question-answering platform, the interactions are represented as $\mathbf{x}_i = (e_i, r_i, t_i)$, where e_i is the exercise that the student attempts at timestamp i and r_t is the correctness of the student's answer. KT aims to predict whether the student will be able to answer the next exercise correctly, i.e., predict $p(r_{t+1} = 1|e_{t+1}, \mathbf{X})$.It is an inherently difficult problem as it is dependent on the factors such as complexity of the human brain and ability to acquire knowledge [6].

It is an important step towards a personalized learning system.

1.2.3 Chapter 5: Interest prediction: Thread Recommendation

Estimating the topics that interest a student at a given time is of importance so that we can build a system to recommend the relevant wikipedia articles, forums or research papers. In this work, we will describe the challenges involved in predicting student interest, how we propose to address those challenges and finally our evaluation scheme. Learners' preferences over MOOC topics evolve as they progress through the course and we attempt to model the student interest so as to recommend them external course material. We developed a model to capture evolving student interest and predict the next forum they will contribute to. Thus, for each student, u, find the most relevant threads that she will be interested in where pth thread is represented as $\mathbf{p}$. Experiments

illustrate that our model outperform the existing thread recommendation baselines.

1.2.4 Chapter 6: Goal Understanding: Learning Trajectory Recommendation

Different students have different learning preferences and identifying these preferences beforehand can help a personalized learning system in designing the learning environment based on their needs. In this work, we propose to understand different student preferences and clustering them based on their interaction with the learning platform. We then devise methods to aid students with different preferences. Essentially, our goal is to help self-learners who enroll in MOOC to master a particular concept (or skill) by providing them the pre-requisite concepts they should master and recommending them the next lecture to watch. To this end, we focus on solving two very important tasks for education community. The first task is to predict the pre-requisite relation between different concepts and second task is to predict the lecture that the user would be interested in to achieve his goal. To solve these tasks, we propose to leverage the knowledge graph that represents the connections and relations between various entities involved in the MOOC platform, the textual content of individual entities and the domain knowledge about concept difficulty. As shown in our experiments, utilizing such rich information results in better representation learning of MOOC entities compared to other representation learning models.

1.3 Bibliographic Statement

Findings of the Chapter 4 on self-attention based models for knowledge tracing have been published in two conference papers [7, 8, 9] and have appeared in (the 12th International Conference on Educational Data Mining, EDM, 2019), and (the 29TH ACM International Conference on Information and Knowledge Management (CIKM) 2020). Contributions made in Chapter 5 has been published in the paper, "Learning Student Interest Trajectory for MOOC Thread Recommendation" and presented at the 20th IEEE International Conference on Data Mining Workshop (ICDMW) 2020 [10]. The generalized version of thread recommendation has been published in the paper "IACN: Influence-aware and Attention-based Co-evolutionary Network for Predicting Dynamic

Embedding" [11] and accepted at the 25th Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD) 2021. In addition, Chapter 6 work has been collected in the article, "MERIT: A Unified Pre-trained Embeddings of MOOC Entities" with preprint available on arXiv.

Chapter 2

Background and Related Work

2.1 Personalized Learning system

Several models have been developed with the goal of personalizing the learning system [12, 13, 14]. A survey on the development of personalized learning was conducted in [12] . This research can classified the research in personalized learning into four categories described in the subsequent paragraph of this section. Adaptive learning technologies include web-based adaptive learning systems and/or intelligent tutoring systems such as developed in [15, 16]. Generally, these systems would analyze student background information, prior knowledge, affective state, preference, and/or learning performance and then algorithmically provide customized learning paths, contents, scaffolds, and/or assessment reports to individual students.

Our system differ from these works from two perspectives: 1) we focus augmenting the current MOOC setting and bring personalization framework on them. We do not intend to build a tutor to teach students but assist self-learners in optimizing their paths by solving the information overload problem. 2) we focus on different respresentation learning methods to build better ML models to solve various tasks important to education community.

2.2 Student Knowledge Modeling

Several models have been developed for modeling student knowledge from their interaction data. These models can be categorized into cognitive diagnosis (from congnitive science) and knowledge tracing (from computation science) domains.

Cognitive models refer to the models designed to discover latent mastery of each student on defined knowledge points. Widely-used approaches could be divided into two categories: one-dimensional models and multi-dimensional models. Among these models, Rasch model [17] (also known as 1PL IRT) is a typical one-dimensional model and computes the probability of getting an exercise correct using logistic regression based on student's ability and exercise (item) difficulty. To improve prediction results, other one-dimensional models include additive factor models [18, 19] which assumed KCs "additively" affect performance. These models include a student's proficiency parameter to account for the variability in student's learning abilities. Comparatively, multi-dimensional models, such as Deterministic Inputs, Noisy-And gate model, characterized students by a binary latent vector which described whether or not she mastered the KCs with the given Q-matrix prior [20].

The KT task evaluates the knowledge state of a student based on her performance data. A Hidden Markov based model, BKT, was proposed in [21]. It models latent knowledge state of a learner as a set of binary variables, each of which represents understanding or non-understanding of a single concept. A Hidden Markov Model (HMM) is used to update the probabilities across each of these binary variables, as a student answers exercises. Further extension of BKT includes, incorporating individual student's prior knowledge [22], slip and guess probabilities for each concept [23] and the difficulty of each exercise [24]. Some approaches [25, 26] use factorization methods to map each student into a latent vector that depicts her knowledge state. To capture the change of student's knowledge evolution over time, [27] proposed a tensor factorization method by adding time as an additional dimension. Another line of research includes methods based on recurrent neural networks such as Deep Knowledge Tracing (DKT) [6], which exploits Long Short Term Memory (LSTM) to model student's knowledge state. Deep Knowledge Tracing plus (DKT+) [28] is an extension of DKT to address the issue faced by DKT such as not being able to reconstruct the input and predicted KCs not

being smooth across the time. Dynamic Key-Value Memory Networks (DKVMN) [29] introduced a Memory Augmented Neural Network [30] to solve KT with key being the exercises practices and values being the mastery of students.

Exercise Relation Modeling: Exercise Relation Modeling has been widely studied in the educational psychology. Some researchers have utilized Q-matrix to map exercises with Knowledge Concepts [31, 20]. Two exercises are related if they belong to the same KC. In addition to Q-matrix based method, recently researchers have started to focus on deriving relations between exercises using the content of exercises. For example, [32, 33, 34] utilize the content of exercises to predict the relation between exercises. After predicting the semantic similarity scores between the exercises [33, 34] use these scores as attention weights to scale the importance of the past interactions. Incorporating exercise relation modeling in KT is an under-explored area. To this end, we explored methods for modeling exercise relations using textual content of exercises and student performance data.

Forget Behavior Modeling: There has been some research exploring the forget behavior of students [35, 36]. *Forget curve theory* introduced in [37] and employed in [36] which claims that student memory decays with time at an exponential rate and the rate of decay is determined by the strength of student cognitive abilities. Recently, DKT-Forget [35] introduce different time-based features in DKT model. DKT-Forgetting considers repeated and sequence time gap, as well as the number of past trials, which is a state-of-the-art method with temporal information. In our work, we take advantage of both exercise relation modeling and forget behavior modeling in KT task which has not been done before.

2.3 Attention Mechanism

Attention mechanism [38] is shown to be effective in tasks involving sequence modeling. The idea behind this mechanism is to focus on *relevant* parts of the input when predicting the output. It makes the models often more interpretable as one can find the weights of specific input that resulted in making a specific prediction. It was introduced for machine translation task to retrieve the words in the input sequence for generating next word in the target sentence. Similarly, it is used in recommendation

systems to predict the next item a person will buy based on his history of purchase. Some models [39, 40] have recognized that augmenting self-attention layer with contextual information improves the performance of the model. Such contextual information include the co-occurrence of items for item recommendation [39] and syntactic and semantic information of a sentence for machine translation [40].

2.4 MOOC Thread Recommendation

The emergence of the internet has led to the development of various discussion platforms for user interaction. Some of the forums (e.g., StackOverflow, and MathExchange) provide a discussion on specific topics while others (e.g., Quora, Reddit, and Yahoo! Answers) can be used to find answers related to a wide variety of topics. Recommendation of forums to the users/experts who can provide insight into a topic and reply to related questions has also been introduced for generic forum recommendation. However, unlike the Community Question Answer (CQA) setting, on MOOC forums, the students are not topic experts, rather learners. The discussion forums on MOOC aims for fostering discussion among students to elevate their understanding through a peer learning experience. Thus, it is important to take into consideration the interest of students while making a recommendation. Student interests evolve as they progress through the course. Additionally, MOOC discussion forums are mostly centered on the topics related to the course. Thus both student's interests and forums can be modeled using topic modeling to obtain distribution over topics.

Joint modeling of users and items, where each interaction between an item and a user updates the state of both interacting user and item has been explored in recommendation systems. RNNs have been used for modeling the evolving features of items and users in [41, 42]. These models, similar to ours, also update the state of users and items after they interact. However, a major difference between these models and our work is that we take into account the course structure to further enhance the performance of our model. Additionally, we project the threads' embeddings personalized to each user such that we can take into account the likelihood of user posting on a thread because of the nature of past posts on the thread.

MOOC has generated a huge amount of data attracting machine learning and data

scientist for research. Here, we discuss all the research done for the recommendation of MOOC discussion forums. The works in [43] use unsupervised topic models with sets of expert-specified course keywords for capturing the category of forum posts. They, then, use topic assignments and sentiment to predict student course completion. Another work [44] analyzed the content of the MOOC forum using topic modeling techniques to automatically generate labels for each thread. These labels can guide students in selecting interesting threads for themselves. Work in [45] couples social network analysis and association rule mining for thread recommendation; while their approach considers social interactions among learners, they ignore the content and timing of posts. The adaptive matrix factorization based method [46] groups learners according to their posting behavior. It also studies the effect of window size, i.e., recommending only threads with posts in a recent time window. The work in [47] uses a rule-based recommendation technique for providing personalized recommendations to individuals. However, these models do not capture the evolving features of users and threads. The point-process based method (PPS) proposed in [48] models the probability that a learner makes a post in a thread at a particular time. This probability is computed based on interest level of the learner on the topic of thread, timescale of the thread topic, timing of the previous posts in the thread, and nature of the earlier posts regarding the learner. However, the user interest on a topic does not remain static across time.

As for modeling temporal dynamics, the work in [49] proposed a method that classifies threads into different categories (e.g., general, technical, social) and ranks thread relevance for learners over time. However, it does not make personalized recommendations since it does not consider learners individually. Another work [50] leverages context trees that are used in a sequential recommendation system for providing adaptive recommendations. MOOC forum recommendation differs from typical sequential recommendation problems because in MOOC forums, both student's interests and threads revolve around the course topics. The course structure is an additional source of information for predicting student interest and expertise.

2.5 Student Profile Learning

Previous work has been conducted to understand learner intent on MOOCs [51, 52, 53]. Most of this work involved designing survey questions to elicit direct response from the learners. This approach, although informative, dos not adequately account for the dynamic nature of learning and interest. To address this issue, we plan to develop machine learning models to infer the intent of learners from their interaction data.

2.6 MOOC Entities representation learning

Real-world education service systems, such as massive open online courses (MOOCs) and online platforms for intelligent tutoring systems on the web offers millions of online courses which have attracted attention from the public [54, 55]. However, the dropout rates on the MOOCs have remained elevated . To maintain student engagement, researchers and practitioners have proposed to personalize the MOOC platform for different profiles of students [56, 57]. In order to build a personalized online learning platform, concept pre-requisite prediction [58, 59] and lecture recommendation [60, 61] have shown to be two important tasks.

2.6.1 Concept Pre-requisite Prediction

Concept pre-requisite prediction is the task of identifying the pre-requisite relations between different concept. Once prerequisite relations among concepts are learned, these relations can be used by self-learners to understand the concepts they have mastered and plan their learning path accordingly. In addition, it can also help the course designers in designing course structure guided by the learned pre-requisites. Previous methods have exploited hand-crafted features and feed it to classification models [59] or investigate active learning with these features to improve classification models [58]. More recently, neural network based methods have been employed to classify pre-requisite relations such as, [62, 63]. Among those [62] utilizes a Siamese network [64] which takes concept representation as input and predicts whether the first concept is pre-requisite of the other. On the other hand, [63] considers the problem of pre-requisite prediction as link prediction task with concepts as nodes of graph and utilizes Graph Variational

Autoencoder [65]. In our work, we pre-train the concept representation using only their text content and the structural information present in course design.

2.6.2 Lecture Recommendation

Lecture recommendation aims at recommending relevant lectures to users based on their historical access behaviors. Such next lecture to watch recommendations as a service has been shown to stimulate and excite learners when they are bored. The overwhelming selection of possible next steps in a MOOC compounded with the complexity of course content can leave a learner frustrated; while, friendly next-step recommendation can be the support they need to move forward and persist [60]. To solve this task, researchers have looked into of various methods. Pardos et. al. [60] modeled the sequence of historic lectures that the student has watched as a sequence and uses sequence encoder to encode the sequence and predict the next lecture of interest. The work in [66] takes as input the sequence of lectures that the student has watched along with the graph connecting various MOOC entities obtained from [5] to recommend the next lecture. This model employs an attention-based graph convolutional networks (GCNs) to learn the representation of different entities. The model discovers user potential interests by propagating users' preferences under the guide of meta-path in the graph. Some studies extracted hidden features from lectures (e.g., textual features extracted from lecture titles, visual features, and acoustic features) and used deep learning methods to learn their representation [67].

Unlike these works, our model learns pre-trained representation of lectures using only the textual content and the course structure. We then use the pre-trained embeddings to improve the downstream tasks. All these applications benefits the development of personalized online learning system [1]. Our work provides a unified representation for MOOC entities compared with previous studies, and provides a solid backbone for applications in the education domain.

2.6.3 Pre-trained Representation Learning in NLP

Recent representation learning methods in NLP rely on training large neural language models on unsupervised data [68, 69, 70, 71, 72, 73]. These methods can be divided

into two categories: feature-based methods, where text is represented by some sort of feature extractors as fixed vectors [72, 73], and pre-training based methods, where parameters of model are pre-trained on corpus and then fine-tuned to specific tasks [70, 68]. Among them, the current state-of-the-art model is BERT [70]. It utilizes Transformer [38] together with some language related pre-training goals, solving many NLP tasks with impressive performance. Although these pre-training solutions have been fully examined in a range of NLP tasks, yet they are not effective to be directly applied to entity representation mainly due to the following three reasons. First, MOOC entities in addition to having textual features have structural relations between each other [5]. These structural relations help in enhancing the embedding quality, would be ignored with these methods that only focus on text [74]. Second, deriving from domain knowledge, the design of courses by an instructor provide information about the entities other than just their linguistic features. Third, off-the-shelf application of these NLP approaches are difficult due to the need of model modification or hyper-parameter tuning, which is inconvenient under many education setup. In addition to that, methods have been developed to encode the relational knowledge in graphs [75]. We use these models to generate embeddings of entities from the Bipartite MOOC graph shown in Figure 5.2 and then use these embeddings as pre-trained embeddings for downstream tasks.

Chapter 3

Vision and Design of POSLS

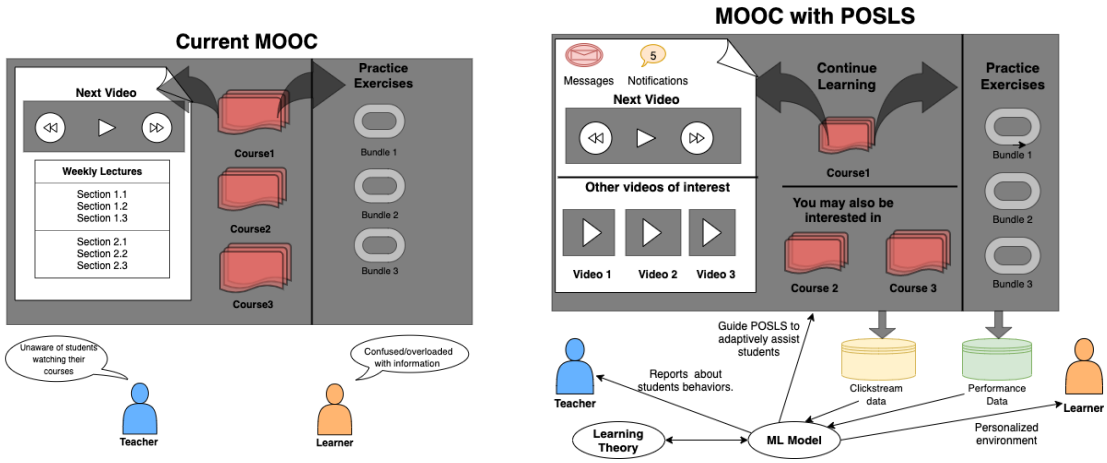
Although online learning have been very popular among the society, it has faced challenges such as high dropout rates, feeling of isolation, standard platform (one size fits all) among others. To tackle these challenges, it is important to build a Personalized Online Self-Learning system. This system is responsible for understanding student needs, interests, strengths and weaknesses and guide them accordingly. This will help accommodate the needs of diverse group of students who enroll on MOOCs and make students feel more connected with the system. Our research is motivated by a number of real-world applications. To provide a viable solution for these problem, the system is required to build models to capture the complexity human learning, their evolving interests and their diverse backgrounds.

3.1 POSLS Vision and Objectives

Current MOOCs do not serve well the large, diverse population of learners attracted to them, with their myriad needs. Unlike traditional *in-class learners* whose primary goal is to complete the course and get credits (since they have paid fees), the goals of *self-learners taking a MOOC class* may fall into a number of distinct categories. Examples include completing the course at regular pace and getting credit (like in-class learners), quickly going through the course as a revision of material they've learnt before, interested only in certain topics to fill-in their knowledge gaps, etc. In addition, self-learners may have very diverse backgrounds from the perspective of preparedness

Table 3.1: POSLS capabilities and their benefits to learners and teachers.

POSLS Capability	**Current Research**	**Proposed Research**	**Benefits to learners**	**Benefits to teachers**
Learner Knowledge Assessment	Teachers assess learner mastery.	Knowledge Tracing model to predict learner mastery.	Inform learners about their strengths and weaknesses.	Informs teachers about different learners' ability.
Comprehensive Personalized Study Material Recommendation	Personalized discussion forums recommendation at an early stage.	Develop a comprehensive recommendation of discussion forums based on both student interest in course topics and learner's past forum activities.	Increases discussions, Addresses information overload, Questions asked on forums get quick answers, Update of other learner activities on MOOC forums, More engagement.	-
Concept Trajectory Recommendation	Recommend list of concepts to self-learners based on their goal.	Incorporate new technologies which specifically utilize massive data to develop better concept relation prediction techniques.	Addresses information overload problem.	-
Course Navigation Behavior Analysis of learners	Sequentially navigate through the whole course or selectively identify relevant course topics.	Understand learner intent to recommend course topics of interest.	Helps learners navigate through the course.	Help teachers modify the course content sequence to align with popular navigation patterns.



(a) Current MOOC setting (b) With POSLS

Figure 3.1: From *Current MOOC* to *MOOC with POSLS*.

for the content of a particular course. Thus, by default, the *one size fits all* approach of organizing the content in current MOOC courses fails to address the needs of its diverse learners.

Personalized Online Self Learning System (POSLS): To address the needs of self-learners, we introduce the concept of *Personalized Online Self Learning (POSL)*, where the pace of learning and the instructional approach implemented are optimized for the needs of each learner [76]. As shown in Figure 3.1, we develop a system that embodies a suite of capabilities to address the needs of different individuals. Essentially the idea is to collect data from POSL platform that will be leverage to build strong ML models. These ML models will personalize the system for individual learners, particularly, recommend the next activity, notify about the forums of interest, and asses each learner knowledge to recommend them the knowledge concept to practice. The course specific modeling layer, consists of techniques to learn representation of content of MOOC (courses, videos, and knowledge concepts). By learning their representations, we can recommend the relevant entities to learners. The idea is to project learners and MOOC entities closer in the same embedding space, such that relevant entities and learners are closer in the space.

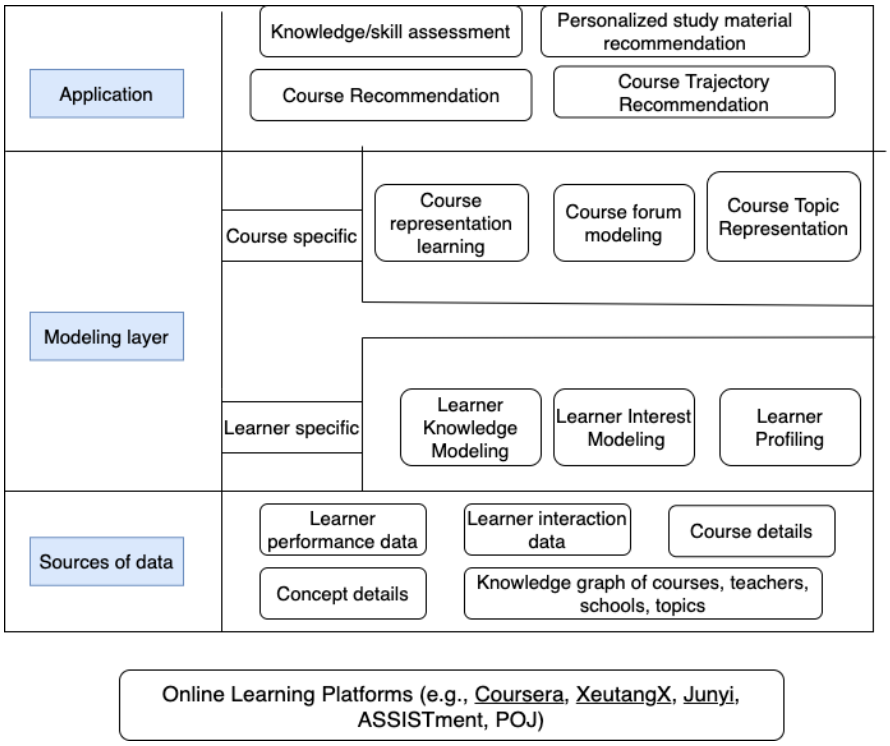


Figure 3.2: Framework for developing a personalized learning system

3.2 POSLS Framework

Figure 3.2 shows the architectural framework of POSLS. As shown, we collect data from various sources such as Coursera [10], XuetangX [5], Junyi [8], ASSISTment and POJ [8]. From these sources, we collect the student performance data, their interaction data (videos watched, forum posts), details of course content (sequence of videos, and topic), universal concept details and a graph linking MOOC entities. We leverage this data to build machine learning models that can provide personalized learning platform (POSLS). Specifically we focus on two different modeling perspectives: student specific and course content specific. The student specific modeling layer focuses on learning student interest representation via their interaction sequences and student knowledge representation via their performance data. We will now describe the major application layer of POSLS.

- **Knowledge/skill assessment.** The first step leading to development of personalized systems is to identify the knowledge of individual learners. In an online education systems, it is important that we determine the strengths and weaknesses of learners before customizing the platform to their condition. A system to assess learner's knowledge can also help in proving a justification to learner regarding what they need to focus or what learning trajectory to follow.

- **Interest Prediction: Recommendation of personalized study material.** MOOCs have become an open source venue for sharing knowledge which generate auxiliary source of learning, such as discussion forums for students. Recent MOOCs (such as, XeutangX) has also released various other sources of information such as micro-blogs, wikipedia, research papers corresponding to each course topics. For students taking the online courses, these auxiliary material can help interested student for acquiring in-depth understanding of course topics or have a constructive discussion with their peers The fundamental challenge involved in this task is that the student interests drift as they progress through the course and it is important to predict the interest of learners at various time instances.

- **Exercise recommendation:** It can be argued that the current MOOC platform infrastructure is designed to efficiently help students 'learn facts' rather than 'acquire skills'. This design is suitable for courses which are heavy on knowledge (including facts, theories and formulae). However, there are courses which emphasize skills (application of knowledge) such as Maths and Computer Programming. For such courses there are many practice platforms such as Junyi [1] , Peking Online Judge (POJ) [2] . However, simply providing a pool of exercises causes information overload for learners. For better utilization, it is important to recommend exercises to the learners. We attempt to utilize the textual information of exercise and learner question-answering sequence to predict which exercise the learner should work on next to maintain its engagement level. Psychology has suggested that the difficulty of student should be smoothly increase so that student's engagement level is kept elevated. To this end, we propose to predict

[1] https://www.junyiacademy.org/
[2] http://poj.org/

difficulty of each exercise by building a pairwise relations between exercises which takes the textual information, knowledge concept associated with the exercise and the question-answering sequence of each student as input.

- **Goal Understanding :Learning Trajectory recommendation.** Some learners, especially professionals, join the online learning system for developing a specific skill (such as machine learning) or learning a particular concept. Under this project we intend to recommend a learning path to these students which consists of next of course topics to pick to meet a particular goal. Since students have different backgrounds, in order to meet a particular goal, different learning paths has to be recommended to different students. In addition, for some courses, topics do not have strong dependence on each other. In such courses the learning path could provide MOOC learners the list of topics they can learn from that course to meet their goals. This will relieve the learners from the task of of going through the entire course material to learn only few topic of their interest [1].

- **Disengagement Prediction : Smart intervention.** A central challenge faced by MOOCs is the extremely high dropout rate — recent reports show that the completion rate in MOOCs is below 5%. It is important to study the factors that cause the users to drop out and their motivation to study MOOCs. With this knowledge student disengagement can be predicted on time and necessary measures can be taken. To improve student retention the intervention mechanism should first identify student who are going to dropout then deploy methods to increase student motivation to complete the course. [77] studied the dropout problem on MOOCs and deployed two intervention mechanism. First is certificate driven in which users receive a message like "Based on our study, the probability of you obtaining a certificate can be increased by about x% for every hour of video watching. Second is effort driven in which user will receive a message to summarize her/his efforts used in this course such as 'You have spent 300 minutes learning and completed 2 homework questions in last week, keep going!"

3.3 Modeling Entities in online learning system

There are various entities in an online learning system : students, courses, course content, exercises, videos, research paper, wikipedia articles. We need to embed these entities such that those entities which are semantically similar are closer in the embedding space. To this end, we define the entities and the associated data we take into account in our proposed framework:

- **Course content and taxonomy** Courses are the foundation of MOOCs and consist of a series of pre-recorded videos. Regarding each course as an entity, we use the synopsis, video list, teacher, and the organization, offering this course as its attributes. Notably, we also consider the description of the teacher and the organization from Wikidata2 as an external resource. In addition to that, we use the mapping of a course with external course material provided in [5]. Specifically, for each video, it records the concept description from Wikidata and search top 10 related papers for each concept via AMiner3 [78, 79, 80] as external resource.

- **Concept graph and relation between courses** Concept graphs [81] refer to the graph consisting of knowledge concepts and each edge define which two concepts are similar or prerequisite of one another. Prerequisite relation has received much attention in recent years and has a direct help for teaching applications. We leverage the prerequisite chains provided by [5] to test our models. It defines relations between different concepts and courses simultaneously.

- **Learners and video watching behavior**: Learner interaction data supports a variety of relevant research (such as course recommendation [82], video navigation [29], dropout prediction), indicates the relationships between courses and concepts [83]. We utilize [5] that logs enrollment records and video watch logs of of learners on the online platform.

- **Learner knowledge and question answering interaction** : Learner interaction data with question-answering platform can help in understanding student strengths and weaknesses. We utilize data from three real-world platforms, Junyi, ASSISTments, and Peking Online Judge provided in [8] to perform evaluation of our knowledge representation model.

Chapter 4

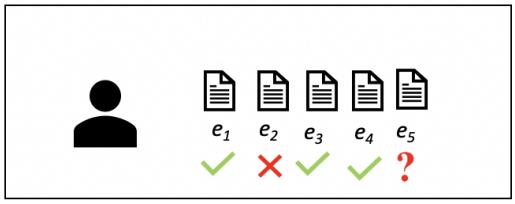
Student Knowledge Modeling

MOOCs requires a mechanism to help students realize their strengths and weaknesses so that they can practice accordingly. In addition to helping students, the mechanism can aid the teachers and system creators to proactively suggest remedial material and recommend exercises based on student needs [84]. For developing such a mechanism, knowledge tracing (KT) is considered to be crucial and is defined as the task of modeling students' *knowledge state* over time [21]. It is an inherently difficult problem as it is dependent on the factors such as complexity of the human brain and ability to acquire knowledge [6].

Figure 4.1 shows an example of a student solving exercises sequentially. When the student encounters a new exercise (e.g. e_5) she applies her knowledge corresponding to the Knowledge Concept (e.g., *Quadratic Equations*) to answer it. The mastery of a particular KC is determined by the past interactions which have a distinct impact on the target KC. The availability of massive dataset of students' learning trajectories about their *knowledge concepts* (KCs), where a KC can be an exercise, a skill or a concept, has attracted data miners to develop tools for predicting students' performance and giving proper feedback [85]. For developing such personalized learning platforms, knowledge tracing (KT) is considered to be an important task and is defined as the task of tracing a student's *knowledge state*, which represents his/her mastery level of KCs, based on his/her past learning activities. The KT task can be formalized as a supervised sequence learning task - given student's past exercise interactions $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t)$, predict some aspect of his/her next interaction $\mathbf{x}_{t+1}$. On the question-answering platform, the

interactions are represented as $\mathbf{x}_t = (e_t, r_t)$, where e_t is the exercise that the student attempts at timestamp t and r_t is the correctness of the student's answer. KT aims to predict whether the student will be able to answer the next exercise correctly, i.e., predict $p(r_{t+1} = 1|e_{t+1}, \mathbf{X})$.

It is an inherently difficult problem as human's ability to solve an exercise is dependent on the complexity of his brain and his knowledge. In order to solve the KT problem, various approaches have been developed. Bayesian Knowledge Tracing (BKT) [21] and its variants [22, 23, 24] model a student's latent knowledge state as a set of binary variables, each of which represents the understanding of a particular KC. A Hidden Markov model, is used to update the latent variables based on the correctness of the observed student opportunities to apply the skill in question. Matrix factorization based methods [**?**] predict the performance of the student on each exercise similar to how rating of an item is predicted in the recommender system. In order to track student's learning process, [27] proposed tensor factorization by incorporating additional time dimension. Performance Factor Analysis (PFA) [18] is a logistic regression based, skill specific method whose regressors are the number of previous correct and incorrect responses on exercises that the student has answered which were questioned on the skills required in the particular exercise. Recently deep learning models such as Deep Knowledge Tracing (DKT) [6] and its variant [28] used Recurrent Neural Network (RNN) to model a student's knowledge state in one summarized hidden vector. The non-linear transitions between input to knowledge state and one knowledge state to the other gives it more flexibility and representational power compared to BKT. Dynamic Key-value memory network (DKVMN) [29] exploited Memory Augmented Neural Network [86] for KT. Using two matrices, *key* and *value*, it learns the correlation between the exercises and the underlying KC and student's knowledge state, respectively. The BKT model and its variants suffer because of their assumptions that each exercise belongs to one concept and representing the mastery of student on each KC with a binary variable, thus neglecting the complexity of human learning [6]. Although the DKT model has shown impressive performance in the KT task, The DKT model faces the issue of its parameters being non-interpretable [87]. DKVMN is more interpretable than DKT as it explicitly maintains a KC representation matrix (*key*) and a knowledge state representation matrix (*value*). However, since all these deep learning models are based on



Exercise tag	Knowledge Concept
e_1	Multiplication
e_2	Square root
e_3	Division
e_4	Linear Equations
e_5	Quadratic equations

Figure 4.1: Left subfigure shows the sequence of exercises that the student attempts and the right subfigure shows the knowledge concepts to which each of the exercises belong.

RNNs, they face the issue of not generalizing while dealing with sparse data [88].

Models such as [89, 90] have shown the importance of explicitly incorporating the relations between KCs as input to the KT model. In particular, [89] uses Dynamic Bayesian Network to model the pre-requisite relations between KCs while [90] incorporate the same in DKT model. However, they assume that the relation between KCs is known apriori. In fact, manual labeling of relations is labor-intensive work. To automatically estimate the relations between exercises, [91] estimates a mapping between each exercise and corresponding KCs and considers the exercise belonging to the same KC as related. While, [34, 33] leverage the textual content of exercises to model semantic similarity relation between exercises. However, these models do not take into account temporal component which affects the importance of past interactions, owing to the dynamic behavior of the student learning process.

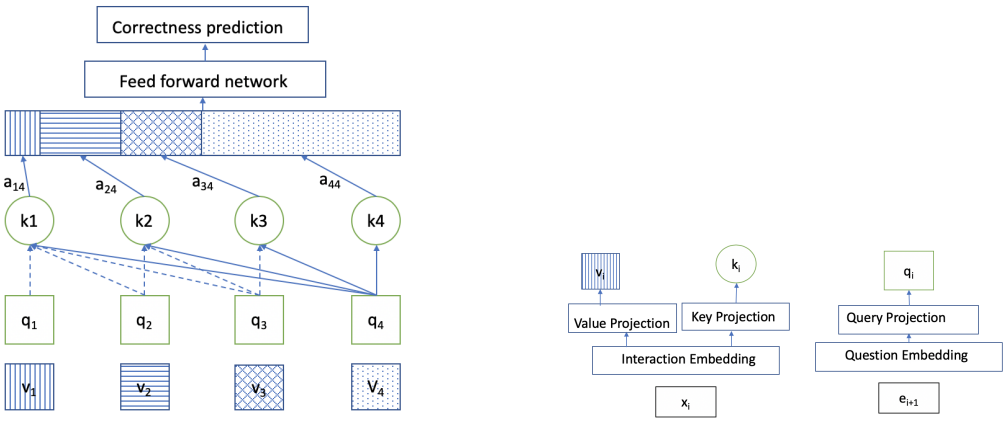
The temporal factors in knowledge tracing have been addressed in [35, 92, 45]. These methods mainly focus on the time elapsed since the last interaction with the same KC or previous interaction without modeling the relation between exercises involved in those interactions. However, as discussed, the previous interactions have a distinct impact on prediction task which is attributed to both exercise relation and temporal dynamics of the learning.

In this work, we propose to use a purely attention mechanism based method, *transformer* [38]. *Transformer* is a sequence modeling method which has shown impressive performance in language modeling. It relies on the fact that each word in a sentence is related to other words and the relevance factor is determined using a compatibility

function between the words. Similarly, In the KT task, the skills that a student builds while going through the sequence of learning activities, are related to each other and the performance on a particular exercise is dependent on his performance on the past exercises related to that exercise. For example, in figure 4.1, for a student to solve an exercise on 'Quadratic equation' (exercise 5) which belongs to the knowledge concept 'Equations', he needs to know how to find 'square roots' (exercise 3) and 'linear equations' (exercise 4). Attention based models proposed in this chapter first identifies *relevant* KCs from the past interactions and then predicts student's performance based on his/her performance on those KCs. For predicting student's performance on an exercise, we used exercises as KCs. As we show later, these models assigns weights to the previously answered exercises, while predicting the performance of the student on a particular exercise. The proposed methods significantly outperform the state-of-the-art KT methods gaining significant performance improvement. Furthermore, the main component (self-attention) is suitable for parallelism; thus, making our model order of magnitude faster than RNN based models. The mathematical notations used in this work are summarized in Table 4.1.

4.1 Theoretical Framework

Central to supporting a learner's learning is estimating with precision what they already know or have learned [93, 94]. This information can provide a learner (or a teacher or a POSLS system) with timely information about the probability that a learner will fail in their next test, so additional learning activities and/or supports can be identified that can prevent this from happening. Awareness of the probability of success is really important in the learning process. Motivation theories [95, 96] predict that expectations to succeed increase engagement, whereas expectation to fail decrease engagement. Estimating learner knowledge is important for another reason. Specifically, it can help identify with precision what a learner does not yet know but can learn with an expert's support, what Vygotsky termed the zone of proximal development (ZPD).



(a) Network of SAKT. At each timestamp the attention weights are estimated for each of the previous element only. Keys, Values and Queries are extracted from the embedding layer shown below. When jth element is query and ith element is key, attention weight is $a_{i,j}$.

(b) Embedding layer embeds the current exercise that the student is attempting and his past interactions. At every time stamp $t+1$, the current question e_{t+1} is embedded in the query space using Exercise embedding and elements of past interactions $\mathbf{x}_t$ is embedded in the key and value space using the Interaction embedding.

Figure 4.2: Diagram showing the architecture of SAKT.

Table 4.1: Notations

Notations	Description
E	total number of exercises
x_i	ith interaction tuple of a student
d	latent vector dimensionality
e	sequence of exercises solved by the student
$\mathbf{P}$	Positional embedding matrix
$\mathbf{A}$	exercise-exercise relation matrix
$\mathbf{R}$	relation coefficients of past interactions
$\hat{\mathbf{x}}_i$	ith interaction embedding
$\mathbf{P}$	Positional embedding matrix
l	maximum sequence length
$\mathbf{E}$	Exercise embedding
$\mathbf{X}$	Interaction sequence of a student:$(x_1, x_2, \ldots, x_i)$

4.2 SAKT: Self-Attentive model for Knowledge Tracing

Our model predicts whether a student will be able to answer the next exercise e_{t+1} based on his previous interaction sequence $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$. As shown in figure 2, we can transform the problem into a sequential modeling problem. It is convenient to consider the model with inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{t-1}$ and the exercise sequence with one position ahead, $e_2, e_3, \ldots, e_t$ and the output being the correctness of the response to exercises $r_2, r_3, \ldots, r_t$. The interaction tuple $\mathbf{x}_t = (e_t, r_t)$ is presented to the model as a number $y_t = e_t + r_t \times E$, where E is the total number of exercises. Thus, the total values that an element in the interaction sequence can take is $2E$, while elements in the exercise sequence can take E possible values.

We now describe the different layers of our architecture.

Embedding layer: We transform the obtained input sequence $\mathbf{y} = (y_1, y_2, \ldots, y_t)$ into $s = (s_1, s_2, \ldots, s_n)$, where n is the maximum length that the model can handle. Since the model can work with inputs of fixed length sequence, if the sequence length, t is less than n, we repetitively add a *padding* of question-answer pair to the left of the

sequence. However, if t is greater than n, we partition the sequence into subsequences of length n. Specifically, when t is greater than n, y_t is partitioned into t/n subsequences each of length n. All these subsequences serve as input to the model.

We train an *Interaction embedding matrix*, $\mathbf{M} \in \mathbb{R}^{2E \times d}$, where d is the latent dimension. This matrix is used to obtain an embedding, $\mathbf{M}_{s_i}$ for each element, s_i in the sequence. Similarly, we train exercise embedding matrix, $\mathbf{E} \in \mathbb{R}^{E \times d}$ such that each exercise in the set e_i is embedded in the e_ith row.

Position Encoding: Position Encoding is the layer in the self-attention neural network which is used for encoding the position so that like convolution network and recurrent neural network, we can encode the order of the sequence. This layer is particularly important in knowledge tracing problem because a student's knowledge state evolves gradually and steadily with time. The output from the embedding layer is embedded interaction input matrix, $\hat{\boldsymbol{M}}$ and embedded exercise matrix, $\hat{\mathbf{E}}$:

$$\hat{\mathbf{M}} = \begin{bmatrix} \mathbf{M}_{s_1} + \mathbf{P}_1 \\ \mathbf{M}_{s_2} + \mathbf{P}_2 \\ \cdots \\ \mathbf{M}_{s_n} + \mathbf{P}_n \end{bmatrix}, \ \hat{\mathbf{E}} = \begin{bmatrix} \mathbf{E}_{s_1} \\ \mathbf{E}_{s_2} \\ \cdots \\ \mathbf{E}_{s_n} \end{bmatrix}. \tag{4.1}$$

Self-attention layer: In our model, we use the scaled dot-product attention mechanism [38]. This layer finds the relative weight corresponding to each of the previously solved exercise for predicting the correctness of the current exercise.

We obtain query and key-value pairs using the following equations:

$$\mathbf{Q} = \hat{\mathbf{E}}\mathbf{W}^Q, \mathbf{K} = \hat{\mathbf{M}}\mathbf{W}^K, \mathbf{V} = \hat{\mathbf{M}}\mathbf{W}^V, \tag{4.2}$$

where $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V \in \mathbb{R}^{d \times d}$ are the query, key and value projection matrices, respectively, which linearly project the respective vectors to different space [38]. The relevance of each of the previous interactions with the current exercise is determined using the attention weights. For finding the attention weights we use the scaled dot product [38], defined as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}. \tag{4.3}$$

Mutiple heads: In order to jointly attend to information from different representative subspaces, we linearly project the queries, keys and values h times using different projection matrices.

$$\text{Multihead}(\hat{\mathbf{M}}, \hat{\mathbf{E}}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O, \tag{4.4}$$

where $\text{head}_i = \text{Attention}(\hat{\mathbf{E}}\mathbf{W}_i^Q, \hat{\mathbf{M}}\mathbf{W}_i^K, \hat{\mathbf{M}}\mathbf{W}_i^V)$ and $\mathbf{W}^O \in \mathbb{R}^{hd \times d}$.
Causality: In our model, we should consider only first t interactions when predicting the result of the $(t+1)$st exercise. Therefore, for a query $\mathbf{Q}_i$, the keys $\mathbf{K}_j$ such that $j > i$ should not be considered. We use, causality layer to mask the weights learned from a future interaction key,

Feed Forward layer: The self-attention layer described above results in weighted sum of values, $\mathbf{V}_i$ of the previous interactions. However the rows of the matrix obtained from the multihead layer, $\mathbf{S} = \text{Multihead}(\hat{\mathbf{M}}, \hat{\mathbf{E}})$ is still a linear combination of the values, $\mathbf{V}_i$ of the previous interactions. To incorporate non-linearity in the model and consider the interactions between different latent dimensions, we use a feed forward network.

$$\mathbf{F} = \text{FFN}(\mathbf{S}) = \text{ReLU}(\mathbf{S}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}, \tag{4.5}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times d}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{d \times d}$, $\mathbf{b}^{(1)} \in \mathbb{R}^d$, $\mathbf{b}^{(2)} \in \mathbb{R}^d$ are parameters learned during training.

Residual Connections: The residual connection [97] are used to propagate the lower layer features to the higher layers. Hence, if low layer features are important for prediction, the residual connection will help in propagating them to the final layers where the predictions are performed. In the context of KT, students attempt exercises belonging to a specific concept to strengthen that concept. Hence, residual connection can help propagating the embeddings of the recently solved exercises to the final layer making it easier for model to leverage the low layer information. A residual connection is applied after both self-attention and feed forward layer.

Layer normalization: In [98], it was shown that normalizing inputs across features can help in stabilizing and accelerating neural networks. We used layer normalization

in our architecture for the same purpose.If $\mathbf{z}$ is the input vector which contains all the features, the operation of layer normalization is defined as:

$$\text{LayerNorm}(\mathbf{z}) = \alpha \odot \frac{\mathbf{z} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{4.6}$$

where $\odot$ is the Hadamard product, α is the μ and σ are the mean and the variance of input vector $\mathbf{z}$. Layer normalization is also applied at both the self-attention and feed forward layer.

Prediction layer:

Finally, each row of the matrix $\mathbf{F}_i$ obtained above is passed through the fully connected network with Sigmoid activation to predict the performance of the student.

$$p_i = \text{Sigmoid}(\mathbf{F}_i\mathbf{w} + \mathbf{b}), \tag{4.7}$$

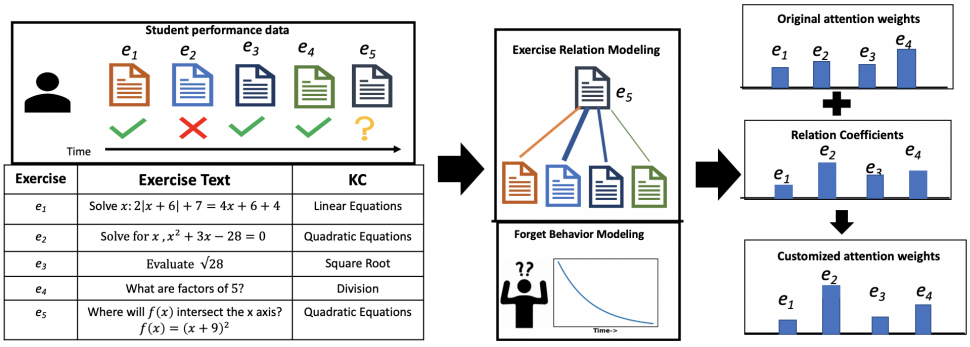
where p_i is a scalar and represents the probability of student providing correct response to exercise e_i, $\mathbf{F}_i$ is the ith row of $\mathbf{F}$ and Sigmoid(z) $= 1/(1 + e^{-z})$

Network Training: The objective of training is to minimize the negative log likelihood of the observed sequence of student responses under the model. The parameters are learned by minimizing the cross entropy loss between p_t and r_t.

$$\mathcal{L} = -\Sigma_t(r_t \log(p_t) + (1 - r_t) \log(1 - p_t)) \tag{4.8}$$

4.3 RKT : Relation-Aware Self-Attention for Knowledge Tracing

SAKT described above is not able to capture the relations between different exercises existing in the exercise pool. In addition, it does not model the forget behavior of learners which present important information about the learners. For example, consider figure 4.3 which shows an example of a student solving exercises sequentially. When the student encounters a new exercise (e.g. 'e_5') she applies her knowledge corresponding to the Knowledge Concept (e.g., *Quadratic Equations*) to answer it. The mastery of a particular KC is determined by the past interactions which have a distinct impact on the target KC. Besides, the impact is distinct under different circumstances. Typically, two



Exercise	Exercise Text	KC
e_1	Solve x: $2\|x+6\|+7=4x+6+4$	Linear Equations
e_2	Solve for x , $x^2+3x-28=0$	Quadratic Equations
e_3	Evaluate $\sqrt{28}$	Square Root
e_4	What are factors of 5?	Division
e_5	Where will $f(x)$ intersect the x axis? $f(x)=(x+9)^2$	Quadratic Equations

Figure 4.3: Overview of RKT model: Leftmost figure shows a student performance data and table shows textual content and knowledge concepts of exercises which constitute the input of RKT. Middle figure shows the relation between exercises and forget behavior of student which serve as contextual information for RKT. Rightmost figure shows that contextual information encoded as relation coefficients informs the attention weight to revised attention weights.

factors account for determining the impact of past interactions in the prediction task: (1) exercise-relation (reflecting the relation between past exercises and the new exercise), and (2) the time elapsed since the past interactions. Intuitively, if the two exercises in the interactions are related to each other then the performance on one affects the other. Additionally, the knowledge gained while solving an exercise in the interaction decays with time, which is attributed to the forget behavior of students. It is important to use this information to contextualize the KT models.

In this sub-chapter, we propose a novel Relation-aware self-attention model for Knowledge Tracing (RKT) that adapts the self-attention
[38] mechanism for KT task. Specifically, we introduce a relation-aware self-attention layer that incorporates the contextual information and meanwhile, maintains the simplicity and flexibility of the self-attention mechanism. To this end, we employ a representation to capture the relation information, called *relation coefficients*. In particular, the *relation coefficients* are obtained from *exercise relation modeling* and *forget behavior modeling*. The former extracts relation between exercises from their textual content and student performance data. While the latter employs a kernel function with a decaying curve with respect to time to model student tendency to forget. Our experiments reveal

that our model outperforms state-of-the-art algorithms on three real-world datasets. Additionally, we conduct a comprehensive ablation study of our model show the effect of key components and visualize the attention weights to qualitatively reveal the model's behavior.

The contribution of this work are:

- We argue that each interaction in the sequence has an adaptive impact on future interaction, where both the relation between the exercises and the forgetting behavior should be taken together into consideration.
- We develop a method to learn the underlying relations between exercises using the textual content and student performance on the exercises which have not been explored before.
- We customize the self-attention model to incorporate the contextual information, thus enabling a fundamental adaptation of the model for KT.
- We perform extensive experiments on three real-world datasets and also illustrate that our model in addition to showing superior performance, provides an explanation for its prediction.

4.3.1 Model Overview

Knowledge Tracing predicts whether a student will be able to answer the next exercise e_n based on his/her previous interaction sequences $X = \{x_1, x_2, \ldots, x_{n-1}\}$. Each interaction is characterized by tuple $x_i = (e_i, r_i, t_i)$, where $e_i \in \{1, \ldots, E\}$ is the exercise attempted, $r_i \in \{0, 1\}$ is the correctness of the student answer, and $t_i \in \mathbb{R}^+$ is the time at which the interaction occurred. For accurate prediction, it is important to identify the underlying relation between e_n attempted at time t_n and the previous interactions. As shown in Figure 4.3 the importance of a past interaction in predicting whether the student will be able to answer the next exercise correctly is determined by two factors: 1) the relation between the exercises solved in the past interaction and the next exercise, and 2) time elapsed since the past interaction. Motivated by this, we develop a Relation-aware Knowledge Tracing model which incorporates the relations as contextual information and propagates it to the attention weights computed using

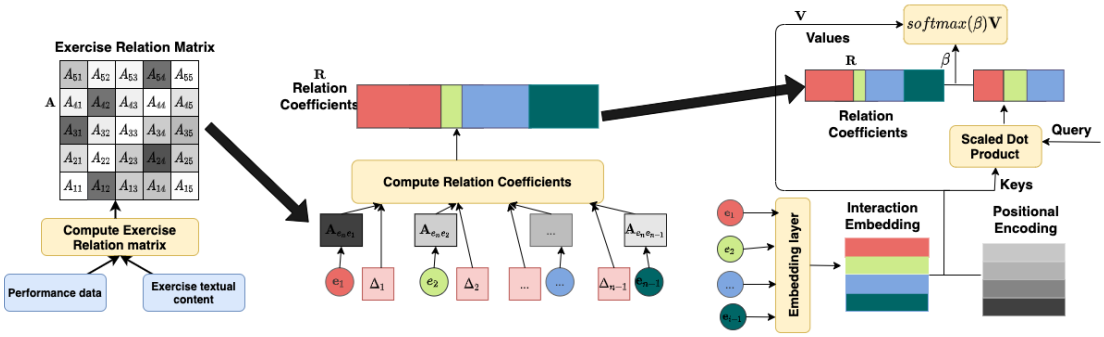


Figure 4.4: The overall architecture of RKT. We first compute the exercise relation matrix **A**. Then we use **A** to compute the relation coefficients based on the relation between past exercises $(e_1, e_2, \ldots e_{n-1})$ and the next exercise e_n and the time elapsed since the interaction $(\Delta_1, \Delta_2, \ldots, \Delta_{n-1})$. The relation coefficients are propagated to the transformer model which modifies the attention weights to take into account the contextual information.

self-attention mechanism [38]. The updated attention weights are then used to compute the weighted sum of the representation of the past interactions which represents the output corresponding to the nth interaction. To learn the parameters, we employ a binary cross entropy loss as our objective function.

We now describe the different layers of our architecture.

4.3.2 Exercise Representation

We learn a semantic representation of each exercise from its textual content. For this, we exploit word embedding technique and learn a function $f : M \rightarrow \mathbb{R}^d$, where M represents the dictionary of words and f is a parameterized function which maps words to d-dimensional distributed vectors. In the look-up layer, exercise content are represented as a matrix of word embeddings. Then the embedding of an exercise i, $\mathbf{E}_i \in \mathbb{R}^d$ is obtained by taking weighted combination of embedding of all the words present in the text of the exercise i using Smooth Inverse Frequency (SIF) [99]. SIF downgrades unimportant words such as but, just, etc., and keeps the information that contributes the most to the semantics of the exercise. Thus, the exercise embedding for an exercise

Table 4.2: A contingency table for two exercises i and j.

		exercise i		
		incorrect	correct	total
exercise j	incorrect	n_{00}	n_{01}	n_{0*}
	correct	n_{10}	n_{11}	n_{1*}
	total	n_{*0}	n_{*1}	n

i is obtained as:

$$\mathbf{E}_i = \frac{1}{|s_i|} \sum_{w \in s_i} \frac{a}{a + p(w)} f(w), \tag{4.9}$$

where a is a trainable parameter, s_i represents the text of ith exercise, and $p(w)$ is the probability of word w.

4.3.3 Exercise-Relation Matrix Computation

An important innovation of our model is that we explore methods of identifying the underlying relations between exercises. Since the relations between exercises are not explicitly known, we first infer these relations from the data and build a *exercise relation matrix*, $\mathbf{A} \in \mathbb{R}^{E \times E}$ such that $\mathbf{A}_{i,j}$ represents the importance that performance on exercise j has on the performance on exercise i. We leverage two sources of information for discovering the relations between exercises: student's performance data and textual content of exercises. The former is used to capture the relevance of knowledge gained in solving exercise j for solving exercise i, while the latter captures the semantic similarity between the two exercises.

We will now describe how learner's performance data can be used to obtain the relevance of the knowledge gained from exercise j to solve exercise i. We first build a contingency table as shown in table 4.2 by considering only the pairs of i and j, where j occurs before i in the learning sequence. If there are multiple occurrences of j in the learning sequence before i, we only consider the latest occurrence. Then, we compute the Phi coefficient which is popularly used as a measure of association for two binary variables. Mathematically the Phi coefficient that describes the relation from j to i is

calculated as,

$$\phi_{i,j} = \frac{n_{11}n_{00} - n_{01}n_{10}}{\sqrt{n_{1*}n_{0*}n_{*1}n_{*0}}}. \tag{4.10}$$

The value of $\phi_{i,j}$ lies between -1 and 1 and a high $\phi_{i,j}$ score means students' performance at j play an important role in deciding their performance at i. We choose Phi coefficients among other correlation metrics to compute the relation between exercises because: 1) it is easy to interpret, and 2) it explicitly penalizes when the two variables are not equal.

Another source of data we use for computing relation between two exercises is the textual content of exercises which informs the semantic similarity of two exercises. We first obtain the exercise embedding of i, $\mathbf{E}_i$ and j, $\mathbf{E}_j$ from section 3.1, then compute the similarity between exercises using cosine similarity of the embeddings. Formally, similarity between exercises is calculated as:

$$\text{sim}_{i,j} = \frac{\mathbf{E}_i\mathbf{E}_j}{||\mathbf{E}_i||_2||\mathbf{E}_j||_2} \tag{4.11}$$

Finally, the relation of exercise j with exercise i is calculated as :

$$\mathbf{A}_{i,j} = \begin{cases} \phi_{i,j} + \text{sim}_{i,j}, & \text{if } \text{sim}_{i,j} + \phi_{i,j} > \theta \\ 0, & \text{otherwise,} \end{cases} \tag{4.12}$$

where θ is a threshold that controls sparsity of relation matrix.

4.3.4 Personalized Relation Modeling

Here we model the contextual information to compute the relevance of past interaction, represented as relation coefficients, for predicting student performance at next exercise. Specifically, we incorporate the exercise relation modeling and forget behavior modeling described below at this step.

Exercise Relation Modeling: This component involves modeling the relation between exercises involved in interaction. Given the past exercises solved by a student, $(e_1, e_2, \ldots, e_{n-1})$ and the next exercise e_n for which we want to predict its performance, we compute the exercise-based relation coefficients from the e_nth row of exercise relation matrix, $\mathbf{A}_{e_n}$ as $\mathbf{R}^E = [\mathbf{A}_{e_n,e_1}, \mathbf{A}_{e_n,e_2}, \ldots, \mathbf{A}_{e_n,e_{n-1}}]$.

Forget behavior modeling: Learning theory has revealed that students forget the knowledge learnt with time [100, 37], known as *forgetting curve theory*, which plays

an important role in knowledge tracing. Naturally, if a student forgets the knowledge gained after a particular interaction i, the relevance of that interaction for predicting student performance at the next interaction should be diminished, irrespective of the relation between exercises involved. The challenge is to identify the interactions whose knowledge the student has forgotten. Since students forget with time, we employ a kernel function that models the importance of interaction with respect to time interval. The kernel function is designed as an exponentially decaying curve with time to reduce the importance of interaction as time interval increases following the idea from forgetting curve theory. Specifically, given the time sequence of interaction of a student $\mathbf{t} = (t_1, t_2, \ldots, t_{n-1})$ and the time at which the student attempts next exercise t_n, we compute the relative time interval between the next interaction and the ith interaction as $\Delta_i = t_n - t_i$. Thus, we compute forget behavior based relation coefficients, $\mathbf{R}^T = [\exp(-\Delta_1/S_u), \exp(-\Delta_2/S_u), \ldots, \exp(-\Delta_{n-1}/S_u)]$, where S_u refers to relative strength of memory of student u and is a trainable parameter in our model.

Following [40], we also obtain revised importance of the past interaction by simply adding the weights obtained from individual sources of information. Thus, we compute the relation coefficients as

$$\mathbf{R} = \text{softmax}(\mathbf{R}^E + \mathbf{R}^T), \tag{4.13}$$

The relation coefficient corresponding to more relevant interaction is higher.

4.3.5 Input Embedding Layer

The raw data of interactions only consists of tuple representing exercise, correctness and time of interaction. We need to embed this information of interactions and positions of interactions. To obtain an embedding of a past interaction j, (e_j, r_j, t_j), we first obtain the corresponding exercise representation using Equation (1). To incorporate the correctness score r_j, we extend it to a feature vector $\mathbf{r}_j = [r_j, r_j, \ldots, r_j] \in \mathbb{R}^d$ and concatenate it to the exercise embedding. Also, we define a positional embedding matrix as $\mathbf{P} \in \mathbb{R}^{l \times 2d}$, to introduce the sequential ordering information of the interactions, where l is the maximum allowed sequence length. The position embedding is particularly important in knowledge tracing problem because a student's knowledge state at a particular time instance should not show wavy transitions [28].

Afterward, we feed the inputs to RKT, and these inputs should convey the representation of interactions and positions in the sequences. Thus, the interaction embedding is obtained as:

$$\hat{\mathbf{x}}_j = [\mathbf{E}_{e_j} \oplus \mathbf{r}_j] + \mathbf{P}_j \tag{4.14}$$

Finally, the input interaction sequence is expressed as $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots \hat{\mathbf{x}}_n]$ by combining the interaction embedding $\mathbf{E}$, and the positional embedding $\mathbf{P}$.

4.3.6 Relation-Aware Self-attention Layer

The core component of RKT is the attention structure that incorporates relation structure. For this, we modify the alignment score of the attention mechanism to attend more to the relevant interactions identified by the relation coefficient, $\mathbf{R}$. Let α be the attention weights learned using scaled dot-product attention mechanism [38] such that

$$\alpha_j = \frac{\exp(e_j)}{\sum_{k=1}^{n-1} \exp(e_k)}, e_j = \frac{\mathbf{E}_{e_n}\mathbf{W}^Q(\hat{\mathbf{x}}_j\mathbf{W}^K)^T}{\sqrt{d}}, \tag{4.15}$$

where $\mathbf{W}^Q \in \mathbb{R}^{d\times d}$ and $\mathbf{W}^K \in \mathbb{R}^{d\times d}$ are projection matrices for query and key, respectively. Finally we combine the attention weights with the relation coefficients, by adding the two weights:

$$\beta_j = \lambda\alpha_j + (1-\lambda)\mathbf{R}_j, \tag{4.16}$$

where $\mathbf{R}_j$ is the jth element of the relation coefficient $\mathbf{R}$. We used addition operation to avoid any significant increase in computation cost. λ is a tunable parameter. The representation of output at the ith interaction, $\mathbf{o} \in \mathbb{R}^d$, is obtained by the weighted sum of linearly transformed interaction embedding and position embedding:

$$\mathbf{o} = \sum_{j=1}^{n-1} \beta_j \hat{\mathbf{x}}_j \mathbf{W}^V, \tag{4.17}$$

where $\mathbf{W}^V \in \mathbb{R}^{d\times d}$ is the projection matrix for value space.

Point-Wise Feed-Forward Layer: We apply the PointWise Feed-Forward Layer (FFN) to the output of RKT by each position. The FFN helps incorporate non-linearity in the model and considers the interactions between different latent dimensions. It consists of two linear transformations with a ReLU nonlinear activation function between the linear transformations. The final output of FFN is $\mathbf{F} = \text{ReLU}(\mathbf{o}\mathbf{W}^{(1)} +$

$\mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$, where $\mathbf{W}^{(1)} \in \mathbb{R}^{d\times d}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{d\times d}$ are weight matrices and $\mathbf{b}^{(1)} \in \mathbb{R}^{d}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^{d\times d}$ are the bias vectors.

Besides of the above modeling structure, we added residual connections [97] after both self-attention layer and Feed forward layer to train a deeper network structure. We also applied the layer normalization [98] and the dropout [101] to the output of each layer, following [38].

4.3.7 Prediction Layer

Finally, to obtain student ability to answer exercise e_n correctly, we pass the learned representation $\mathbf{F}$ obtained above through the fully connected network with Sigmoid activation to predict the performance of the student.

$$p = \sigma(\mathbf{F}\mathbf{W} + \mathbf{b}), \tag{4.18}$$

where p is a scalar and represents the probability of student providing correct response to exercise e_n, and $\sigma(z) = 1/(1 + e^{-z})$.

4.3.8 Network Training

Since the self-attention model works with sequence of fixed length, we convert the input sequence, $X = (x_1, x_2, \ldots, x_{|X|})$, into sequence of fixed length l before feeding it to RKT. If the sequence length, $|X|$ is less than l, we repetitively add a *padding* to the left of the sequence. However, if $|X|$ is greater than l, we partition the sequence into subsequences of length l. The objective of training is to minimize the negative log likelihood of the observed sequence of student responses under the model. The parameters are learned by minimizing the cross entropy loss between p and r at every interaction.

$$\mathcal{L} = -\sum_{i\in I}(r_i \log(p_i) + (1 - r_i)\log(1 - p_i)), \tag{4.19}$$

where I denotes all the interactions in the training set.

Table 4.3: Dataset Details

	ASSIST2012	Junyi	POJ
# students	39,364	238,120	22,916
# exercises	58,761	684	2,751
# Interactions	4,193,631	26,666,117	996,240
Avg exercise record/student	107	111.99	43.47
Duration of data collection	365 days	1095 days	258 days

4.4 Experimental Settings

In this section, we present our experimental settings to answer the following questions:
RQ1 Can attention-based models outperform the state-of-the-art methods for Knowledge Tracing?
RQ2: What is the influence of various components in the RKT and SAKT architecture?
RQ3 Are the attention weights able to learn meaningful patterns in computing the embeddings?

4.4.1 Datasets

To evaluate our model, we used three real-world datasets.

- **ASSISTment2012(ASSIST2012)**[1] **:** This dataset is provided by ASSISTment online tutoring platform and is widely used for KT tasks. We also utilized the problem bodies to conduct our experiments.

- **JunyiAcademy (Junyi)**[2] This dataset was collected by JunyiAcademy[3] in 2015 [102]. The available dataset only contains the exercising records of students. To obtain the textual content we scraped the data from their website. Overall,

[1] **https://sites.google.com/site/assistmentsdata/home/2012-13-school-data-with-affect**
[2] **https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=1275**
[3] https://www.junyiacademy.org/

this dataset contains 838 distinct exercises and we removed exercises which do not contain textual content.

- **Peking Online Judge (POJ)** This dataset is collected from Peking online platform of coding practices and consists of computer programming questions.We scraped the publicly available data from the website [4] .

For all these datasets, we first removed the students who attempted fewer than two exercises and then removed those exercises which were attempted by fewer than two students. The complete statistical information for all the datasets can be found in Table 6.2. The code and dataset is available at *https://github.com/shalini1194/RKT*.

4.4.2 Implementation Details

Word Embeddings

The first step in our method is to embed exercise content and initializing each word of the exercise content. All exercises are truncated to no more than 200 words. However, mathematical exercises consists of words not found in traditional English articles such as, news. For example it is common to find formulas like "$\sqrt{(}x) + 1$" in mathematical exercise which carry important information about the exercise. Therefore, to preserve the mathematics semantics, we transform each formula into its TEX code features ("$\sqrt{(}x) + 1$" is transformed to " sqrt x + 1"). After initialization, each exercise is represented with sequence with vocabulary words and TEX tokens. The model is trained by embedding each word into an embedding vector with 50 dimensions (i.e., d = 50) by using word2vec [5] .

Framework Setting

We now specify the network initializations in our model. We set the model dimension in self-attention as 64 and the maximum allowed sequence length l as 50. The model is trained with a mini-batch size of 128. We use Adam optimizer with a learning rate

[4] http://poj.org/
[5] https://radimrehurek.com/gensim/models/word2vec.html

of 0.001. The dropout rate is set to 0.1 to reduce overfitting. The L2 weight decay is set to 0.00001. All the model parameters are normally initialized with 0 mean and 0.01 standard deviation. The value of sparcity controlling threshold, θ used in Eq. (4.12) is 0.8 in our experiments. We trained the model with 80% of the dataset and test it on the remaining. We perform 5-fold cross validation to evaluate all the models, in which folds are split based on students.

4.4.3 Metrics

The prediction of student performance is considered in a binary classification setting i.e., answering an exercise correctly or not. Hence, we compare the performance using the Area Under Curve (AUC) and Accuracy (ACC) metric. Similar to evaluation procedure employed in [35, 6], we train the model with the interactions in the training phase and during the testing phase, we update the model after each exercise response is received. The updated model is then used to perform the prediction on the next exercise. Generally, the value 0.5 of AUC or ACC represents the performance prediction result by randomly guessing, and the larger, the better.

4.4.4 Approaches

Knowledge Tracing (KT)

We compare our model against the state-of-the-art KT methods.

- **DKT** [6] : This is a seminal method that uses single layer LSTM model to predict the student's performance. In our implementation of DKT, we used norm-clipping and early stopping to improve the performance as has been employed in [29].
- **DKVMN** [29]: This is a Memory Augmented Recurrent Neural Network based method where in the relation between different KCs are represented by the *key* matrix and the student's mastery of each KC by the *value* matrix.
- **DKT+Forget** [35] : This is an extension of DKT method which predicts student performance using both the student's learning sequence and fogetting behavior.
- **EERNN** [34]: This model utilizes both the textual content of exercises and student's exercising records to predict student performance. They use RNN as

Table 4.4: Performance comparison. The best performing method is boldfaced, and the second best method in each row is underlined. Gains are shown in the last row.

	ASSIST2012		POJ		Junyi	
	AUC	ACC	AUC	ACC	AUC	ACC
DKT	0.712	0.679	0.656	0.691	0.814	0.744
DKVMN	0.701	0.686	0.704	0.700	0.822	0.751
DKT+Forget	0.722	0.685	0.662	0.700	0.840	0.759
EERNN	0.748	0.698	0.733	0.720	0.837	0.758
EKT	<u>0.754</u>	<u>0.702</u>	<u>0.737</u>	<u>0.729</u>	<u>0.842</u>	<u>0.759</u>
SAKT	0.735	0.692	0.696	0.705	0.834	0.757
RKT	**0.793**	**0.719**	**0.827**	**0.774**	**0.860**	**0.770**
Gain%	5.172	2.422	12.212	6.173	1.775	1.050

the underlying model to learn the exercise embedding and the student knowledge representation. Furthermore, they attend over the past interactions using the cosine similarity between the past interactions and the next exercise.

- **EKT** [33]: This model is an extension of the EERNN model which also tracks student knowledge acquisition on multiple skills. Specifically, it models the relation between the underlying Knowledge Concepts to enhance the EERNN model.

4.5 Results and Discussion

4.5.1 Student Performance Prediction (RQ1)

Table 4.4 shows the performance of all baseline methods and our RKT model. We have the following observations:

Different kinds of baselines demonstrate noticeable performance gaps. SAKT model shows improvement over DKT and DKVMN model which can be traced to the fact that SAKT identifies the relevance between past interactions and next exercise. DKT-Forget further gains improvements most of the time, which demonstrates the importance

of taking temporal factors into consideration. Further, EERNN and EKT incorporate textual content of exercises to identify which interaction history is more relevant and hence perform better than the those models which do not take into account these relations. RKT performs consistently better than all the baselines. Compared with other baselines, RKT is able to explicitly captures the relations between exercises based on student performance data and text content. Additionally, it models learner forget behavior using a kernel function which is more interpretable and proven way to model human memory [37] compared to DKT+forget model.

Second, the performance gain is lowest for Junyi dataset. We believe that a possible reason of low improvement on Junyi is that since the number of exercises in Junyi is fairly small the relation between exercises can be modeled by sequential models such as RNN and self-attention mechanism. It does not need explicit relation learning based on the content.

We would also like to point out that, combining the model with contextual information in RKT does not lead to any significant increase in runtime of the model and it remains as scalable as SAKT model. SAKT and RKT are more scalable than other sequential models because of its parallelization capability [7]. The performance gain on POJ dataset is the best compared to the other datasets. This can be attributed to the fact that on POJ dataset is the datset obtained from online judge where students solve questions on different topics randomly. On the other hand, on ASSISTment and Junyi platforms the questions consists of High School Maths dataset where students generally follow a certain path set by the instructor. Thus, the the role of relations between questions plays an important role in POJ dataset compared to others.

Performance comparison w.r.t. interaction sparsity

One benefit of exploiting the relations between interactions is that it makes our model robust towards sparsity of dataset. Exploiting the relation between different exercises can help in estimating student performance at related exercises, thus alleviating the sparsity issue.

To verify this, we perform an experiment over student groups with different number of interactions. In particular, we generate four groups of students based on interaction number per user, thus generating groups with less than 10, 100, 1000, 10000 interactions,

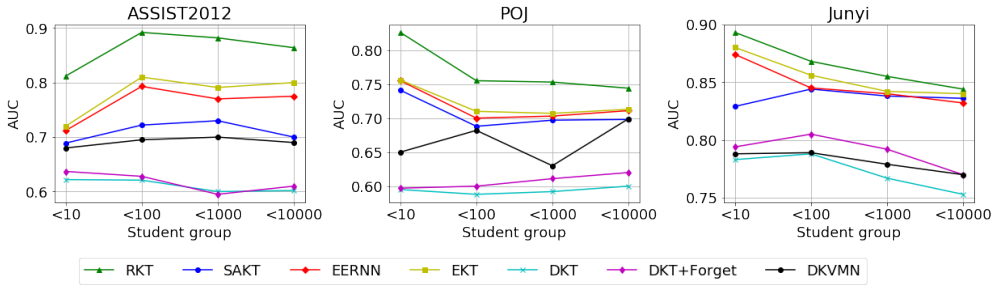


Figure 4.5: Plot of prediction performance over different student groups based on sparsity of interaction levels. Our model, RKT significantly outperforms every baseline.

Table 4.5: Ablation Study of RKT

	ASSIST2012		POJ		Junyi	
	AUC	ACC	AUC	ACC	AUC	ACC
PE	0.788	0.712	0.790	0.749	0.848	0.763
TE	0.787	0.712	0.816	0.766	0.835	0.758
RE	0.755	0.696	0.686	0.710	0.835	0.763
PE+TE	0.778	0.705	0.788	0.746	0.833	0.754
PE+RE	0.759	0.699	0.676	0.700	0.832	0.757
RE+TE	0.735	0.692	0.696	0.705	0.834	0.757
PE+RE+TE	0.730	0.684	0.667	0.693	0.830	0.756
RKT	**0.793**	**0.719**	**0.827**	**0.774**	**0.860**	**0.770**

respectively. The performance of all the methods is displayed in Figure 4.5. We find that RKT outperforms the baseline models in all the cases, signifying the importance of leveraging relation information for predicting performance. Also, the performance gain of RKT for student groups with less number of interactions is more significant. Thus, we can reach to a conclusion that RKT which exploits the relation between interactions is effective for learning knowledge representation of students even with less interactions.

4.5.2 Ablation Study (RQ2)

To get deep insights on the RKT model, we investigate the contribution of various components involved in the model. Therefore, we conduct some ablation experiments to show how each part of our method affect final results. In Table 4.5, there are seven variations of RKT, each of which takes out one or more opponents from the full model. Specifically:PE, TE, RE refer to RKT without position encoding, forget behavior modeling and exercise relation modeling, respectively. PE+TE, PE+RE, TE+RE refer to removal two components simultaneously, i.e. position encoding and forget behavior modeling, position encoding and exercise relation modeling, and exercise relation modeling and forget behavior modeling, respectively. And finally, PE+RE+TE refers to RKT that does not model the position encoding, forget behavior modeling and exercise relation modeling for interaction representation. The result in Table 4.5 indeed shows many interesting conclusions.

First, the more information a model encodes, the better the performance, which agrees with the intuition. Second for all datasets removing exercise relation modeling causes the most drastic drop in performance. This validates our argument that explicitly learning exercise relations is important for improving the performance of KT model. Thirdly, incorporating the forget behavior model in RKT which of students causes more improvement in ASSIST2012 and Junyi datasets than POJ. We hypothesize that this can be attributed to the fact that the concepts involved in solving POJ exercises are less diverse than those involved in high school maths course (Junyi and ASSIST2012 dataset). As a result in majority cases the reason of wrong answer on POJ is the confusion or true knowledge gaps in the students, rather than their forgetting behavior.

Effect of Exercise Relation matrix computation

To explore the impact of exercise relation matrix computation, we consider the variants of RKT that uses different settings. We explore the following methods for computing exercise relation matrix:

1. Previous work such as [91, 103], considered that two exercises are related if they belong to the same KC. We also employ this technique and build an exercise

Table 4.6: Comparison of four exercise relation matrix computation methods.

	ASSIST2012		POJ		Junyi	
	AUC	ACC	AUC	ACC	AUC	ACC
Method (1)	0.755	0.700	-	-	0.764	0.710
Method (2)	0.782	0.708	0.755	0.733	0.836	0.759
Method (3)	0.785	0.709	0.763	0.737	0.844	0.762
Method (4)	**0.793**	**0.719**	**0.827**	**0.774**	**0.860**	**0.770**

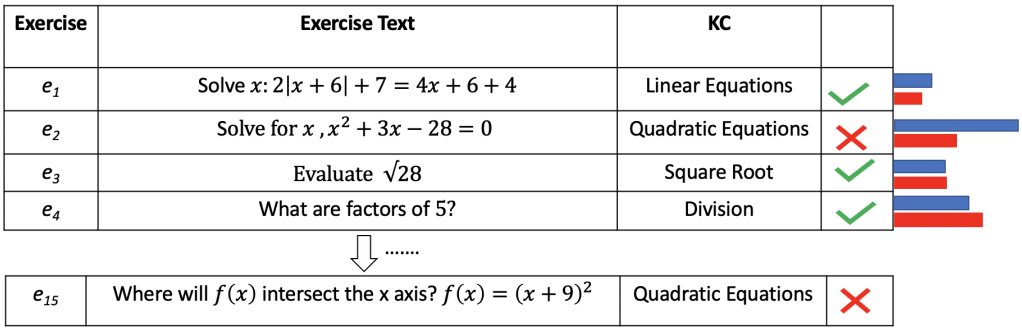


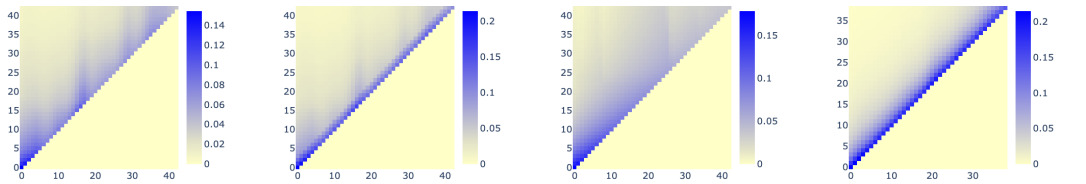
Figure 4.6: Attention visualization in RKT model of an example student from Junyi. We predict her performance on e_{15} based on her past 15 interaction (we only show the first 4 interactions for better illustration). Right bars show the attention weights of two RKT (blue) and SAKT (red)

relation matrix with boolean values such that $\mathbf{A}_{i,j} = 1$ if i and j belong to the same KC otherwise 0.

2. Use only the textual content of two exercises to estimate the relation between them. We compute the relation between two exercises with Equation (3) only.

3. Use the student performance data to compute the relation between two exercises. Only Equation (2) is employed to compute the relation between two exercises.

4. Use both textual content and student performance data to compute the similarity between two exercises. We compute the relation coefficients using Equation (4).



(a) ASSIST2012 - SAKT (b) ASSIST2012 (c) POJ (d) Junyi

Figure 4.7: Visualization of attention weights on different datasets. Each subfloat depicts the average attention weights of different sequences of the corresponding datasets.

We do not have information about the exercise-to-KC mapping for POJ data and hence can not apply method (1) for POJ. Specifically Table 4.6 summarizes the experimental results. The findings are:
Firstly, Method (1) performs the worst among all the four methods. This can be attributed to the fact that linking exercises only based KCs ignores the fact there exists relation among exercises which do not belong to the same KC. Method (3) also shows performance gain over method (2) as student performance data is a good indicator of how relations between exercises are perceived by the students. Even if textual content of two exercises are not similar the association of knowledge involved in solving the two exercises could be high. Finally, method (4) that leverages both student performance data and exercise textual content data outperforms the other methods.

4.5.3 Attention weights visualization (RQ3)

Benefiting from a purely attention mechanism, RKT and SAKT models are highly interpretable for explaining the prediction result. To this end, we compared the attention weights obtained from both these models. We selected one student from Junyi dataset and obtain the attention weights corresponding to the past interactions for predicting her performance at exercise e_{15}. Figure 4.11 shows the weights assigned by both SAKT and RKT. We see that compared to SAKT, RKT places more weights on e_2 which belongs to same KC as e_{15} and have stronger relation. Since the student gave wrong

answer to e_2, she has not yet mastered "Quadratic Equations". As a result, RKT predicts that the student will not be able to answer e_{15}. Thus, it is beneficial to consider relations between exercises for KT.

We also performed experiment to visualize the attention weights assigned by RKT on different datasets. Recall that at time step t_i, the relation-aware self-attention layer in our model revise the attention weights on the previous interactions depending on the time elapsed since the interaction and the relations between the exercises involved. To this end, we examine all sequences and seek to reveal meaningful patterns by showing the average attention weights on the previous interactions.

Figure 4.7 shows the heatmap of attention weight matrix where (i, j)th element represents the attention weight on jth element when predicting performance at ith interaction. Note that when we calculate the average weight, the denominator is the number of valid weights, so as to avoid the influence of padding for short sequences. We consider a few comparisons among the heatmaps:

- (b), (c), (d): The heatmap representing the attention weights pertaining to different datasets reveals that recent interactions are given the higher weights compared to other interaction. It can be attributed to the forget behavior of learning process such that only the recent interactions can inform the student knowledge state.

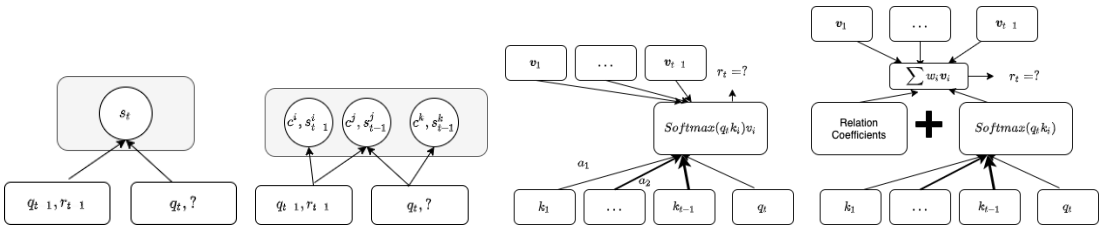
- (b) vs. (c): This comparison shows the weights assigned by RKT on two different types of dataset. In ASSIST2012 dataset, the exercises are sequenced for *skill-building*, i.e., they are organized so that a student can master one skill first and then learn the next skill. As a result in ASSIST2012 the exercises adjacent to each other are related. While, in POJ dataset, student chooses exercises based on their needs. As a result, the heatmap corresponding to ASSIST2012 dataset has attention weights concentrated towards the diagonal elements, while for POJ the attention weights are spread across the interactions.

- (a) vs. (b): This comparison shows the effect of relation information for revising the attention weights. Without relation information the attention weights are more distributed over previous interaction, while the relation information concentrates the attention weights closer to diagonal as adjacent interactions in ASSIST2012 have higher relations.

4.6 KT models on Large-Scale Dataset

In this section, we perform an analysis of the described deep-learning models for knowledge tracing. This analysis will help understand which deep-learning model performs best when we have massive student performance dataset. This work has been published in [9]. In addition, we visualize the attention weights to qualitatively reveal SAKT and RKT behavior.

To summarize, figure 4.8 represents the difference between the four models we have analyzed in this work. First DKT uses a summarized hidden vector to model the knowledge state. Second, DKVMN maintains the concept state for each concept simultaneously and all concept states constitute the knowledge state of a student. Third, SAKT assigns weights to the past interaction using self-attention mechanism to identify the relevant ones. It then uses the weighted combination of these past interactions to estimate student knowledge on the involved KCs and predict her performance. Finally, RKT improves over SAKT by introducing a relation coefficient added to the attention weights learned from SAKT. The relation coefficient are learned from the contextual information explicitly modelling the relation between exercises involved in the past interactions and student forget behavior of students.



(a) DKT (b) DKVMN (c) SAKT (d) RKT

Figure 4.8: Model differences among DKT, DKVMN, SAKT and RKT.

4.6.1 Data

To compare the deep-learning methods for KT, we use large-scale student interaction dataset, EdNet released in [104]. EdNet consists of all student-system interactions collected over a period spanning two years by Santa, a multi-platform AI tutoring

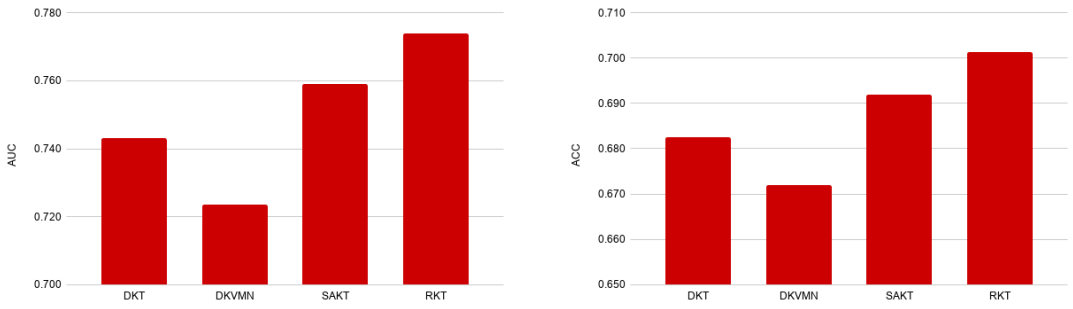
service with approximately 780,000 students. It has collected a total of 131,441,538 student interactions with each student generating an average of 441.20 interactions. The dataser consists a total 13,169 problems and 1,021 lectures tagged with 293 types of skills, and each of them has been consumed 95,294,926 times and 601,805 times, respectively.

4.6.2 Evaluation Setting

The prediction of student performance is considered in a binary classification setting i.e., answering an exercise correctly or not. Hence, we compare the performance using the Area Under Curve (AUC) and Accuracy (ACC) metric. Similar to evaluation procedure employed in [35, 6], we train the model with the interactions in the training phase and during the testing phase, we update the model after each exercise response is received. The updated model is then used to perform the prediction on the next exercise. Generally, the value 0.5 of AUC or ACC represents the performance prediction result by randomly guessing, and the larger, the better.

To ensure fair comparison, all models are trained with embeddings of size 200. The maximum allowed sequence length for self-attention is set as 50. The model is trained with mini-batch size of 128. We use Adam optimizer with a learning rate of 0.001. The dropout rate is set to 0.1 to reduce overfitting. The L2 weight decay is set to 0.00001.

4.6.3 Results and Discussions



(a) AUC (b) ACC

Figure 4.9: Performance Comparison. RKT performs best among the models.

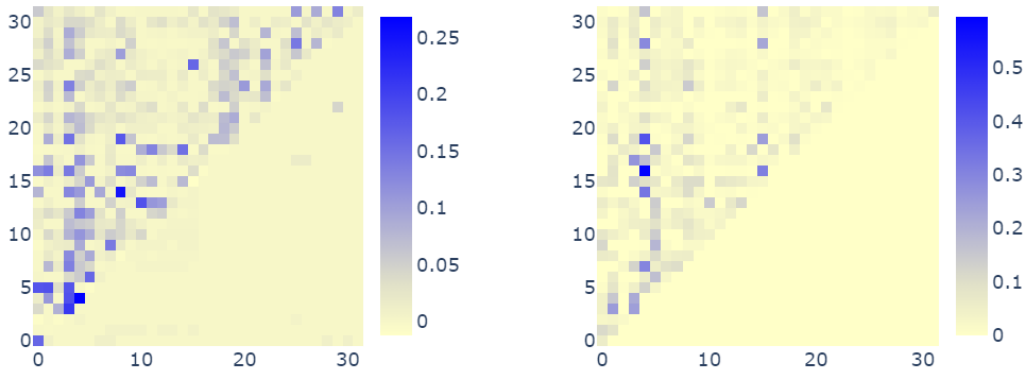
Quantitative Results

Figure 4.9 shows the performance comparison of deep-learning models for KT on Ednet dataset. Different kinds of baselines demonstrate noticeable performance gaps. SAKT model shows improvement over DKT and DKVMN model which can be traced to the fact that SAKT identifies the relevance between past interactions and next exercise. RKT performs consistently better than all the baselines. Compared with other baselines, RKT is able to explicitly captures the relations between exercises based on student performance data and text content. Additionally, it models learner forget behavior using a kernel function which is more interpretable and proven way to model human memory [37].

The results reveal that provided enough data, attention-based models surpass the other sequence encoder techniques such as RNN, LSTM and Memory Augmented Networks. Furthermore, incorporating contextual data such as relation between exercises and domain knowledge such as student forget behavior attribute to performance gain even after availability of the massive dataset. This motivates us to further explore Knowledge Guided Machine Learning in the KT task.

Qualitative Analysis

Benefiting from a purely attention mechanism, RKT and SAKT models are highly interpretable for explaining the prediction result. Such interpetability can help understand which past interactions played an important role in predicting student performance on the next exercise. To this end, we compared the attention weights obtained from both RKT and SAKT. We selected one student from the dataset and obtain the attention weights corresponding to the past interactions for predicting her performance at an exercise. Figure 4.10 shows the heatmap of attention weight matrix where (i, j)th element represents the attention weight on jth element when predicting performance on ith interaction. We compare the generated heatmap for both SAKT and RKT. This comparison shows the effect of relation information for revising the attention weights. Without relation information the attention weights are more distributed over previous interaction, while the relation information concentrates the attention weights to specific relevant interactions.

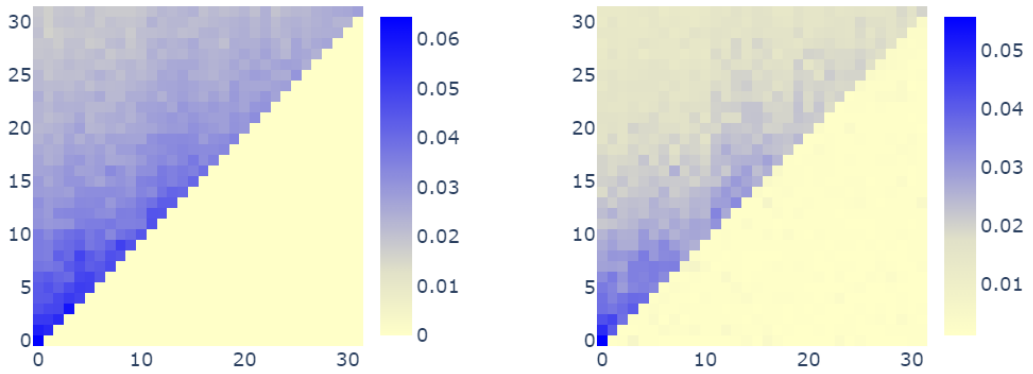


(a) SAKT (b) RKT

Figure 4.10: Visualization of attention weights of an example student from EdNet by SAKT and RKT. Each subfloat depicts the attention weights assigned by the models for that student.

Finally we also performed experiment to visualize the attention weights averaged over multiple sequences by RKT and SAKT. Recall that at time step t_i, the relation-aware self-attention layer in our model revises the attention weights on the previous interactions depending on the time elapsed since the interaction, and the relations between the exercises involved. To this end, we examine all sequences and seek to reveal meaningful patterns by showing the average attention weights on the previous interactions. Note that when we calculate the average weight, the denominator is the number of valid weights, so as to avoid the influence of padding for short sequences. Figure 4.11 compares average attention weights assigned by SAKT and RKT. This comparison shows the effect of relation information for revising the attention weights. Without relation information the attention weights are more distributed over previous interaction, while the relation information concentrates the attention weights closer to diagonal.Thus, it is beneficial to consider relations between exercises for KT.

We present the concluding remarks in Chapter 7



(a) SAKT

(b) RKT

Figure 4.11: Visualization of attention weights pattern on different datasets. Each subfloat depicts the average attention weights of different sequences.

Chapter 5

Interest prediction: Thread Recommendation

Massive Open Online Courses (MOOCs) provide a platform to teach students all kinds of subjects or courses. However, learning from MOOC comes with its own set of challenges. One of the unique challenges faced by online learning platforms is that the means of interaction between students and instructors are critically limited. Peer learning, i.e., learning from each other through discussion, is an important component of the learning procedure and has a positive impact on student learning. On MOOCs, discussion forums facilitate peer learning where instructors and students can ask questions, discuss ideas, and provide help to other students. It has been shown that more the learner's participation on MOOC forums results in higher performance gains [105]. However, as class size grows, the number of forums per course increases rapidly. As a result, it becomes quite difficult for a student to filter through a vast and overwhelming number of open forums to find relevant threads.

To address the information overload problem discussed above, it is necessary to build a thread recommendation system that yields a personalized shortlist of threads based on the student interest. Furthermore, the thread recommendation system in MOOCs helps decrease the amount of time required for new questions to go unanswered by directing appropriate users there [46]. Traditional recommendation models have been used for

recommending threads using collaborative filtering [47] and adaptive matrix factorization [46]. However, certain characteristics of thread recommendation on MOOCs set them apart from traditional recommendation systems. Firstly, MOOC forums are frequently updated by students or the instructors, which diversifies the content of these forums. Simultaneously, learners' preferences over MOOC topics evolve as they progress through the course; yet traditional recommendation techniques assume that learner interests and thread properties are static [106]. To capture this dynamic nature of the MOOC thread recommendation, a sequential recommendation model based on context tree [107] was proposed in [50]. However, the main issue with such sequential recommendation models is that the student interest representation is updated only when an action (a reply on a thread or a post) occurs. However, *a student interest in a topic keeps evolving even when it has not taken any action.*

To tackle this, our work Student Interest Trajectory based Recommendation (SITRec) represents students and thread as embedding vectors. The evolution of student(and thread) is captured by a sequence of learned embeddings, which represents a trajectory of a student interests(and thread properties). Two key operations are employed to learn this trajectory: *update operation* and *projection operation*. The update operation updates the embedding of a student and a thread whenever an action involving the two is observed. It employs two mutually-recursive Recurrent Neural Networks (RNNs). One of them updates the student embedding using the thread embedding, and the other updates the thread embedding using the student embedding.

Furthermore, even in the absence of any action, we update the embedding of students and threads using the projection operation. The projection operation consists of two components: student projection operation and thread projection operation. The student projection operation is designed based on two intuitions. Firstly, as more time elapses time since student's last update, her embedding will get farther. Secondly, the course topic that a student is studying at a time is a good indicator of her interest, and course topics are sequenced as defined in the course structure. Thus, incorporating the course topic as a context feature in projection operation is beneficial in learning her projected embedding. The thread projection operation learns personalized thread embedding for each student. Intuitively, student interest in a thread further increases, by different factors, if another student posts on it after the student's post or provide

explicit comments to the student's post [48]. As a result, thread projection operation projects thread embedding with respect to student embedding based on nature of posts made on it after the student's post.

To predict the next thread which the student will be interested in, our model predicts embedding of the next thread. The recommendations can be made via nearest-neighbor search centered at that predicted embedding.

Extensive experimentation on real-world datasets shows that SITRec significantly outperforms the existing thread recommendation and dynamic embedding methods on Mean Average Precision (MAP). We conduct a comprehensive ablation study to show the effect of key components and visualize the drift in student interest and how it can be leveraged to find the topic of interest for each student. Summary of major contributions of our work are:

- We consider the problem of thread recommendation as a dynamic sequential recommendation where both student embeddings and the thread embeddings keep evolving. We model the inter-dependency between the evolution of student and thread using mutually-recursive RNNs.
- We propose to predict student interest at a future time and then extract the relevant threads. We propose to utilize the course topic that the student is studying and elapsed time to predict the future interest of the student.
- We propose to project thread embedding personalized for each student so that we can incorporate how the interest of a student in a thread changes with the nature of posts made on the thread.
- We performed extensive experimentation involving an ablation study and visualizing the drift in student interest to support our methodology.

5.1 Theoretical Framework

Motivation theories emphasize situational interest as an integral component to learner motivation, and thus engagement [**?**, 108]. Situational interest is a psychological state that arises through interactions with learning tasks and can fluctuate over the course of

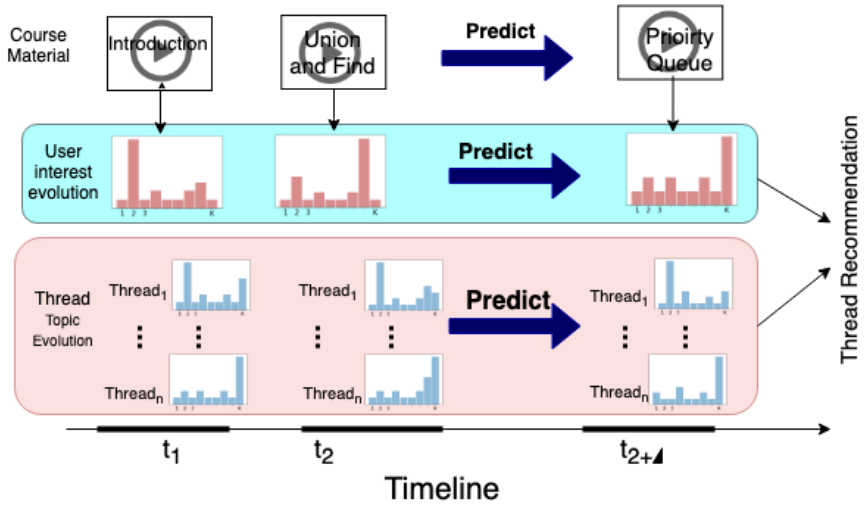


Figure 5.1: Temporal evolution of student interest and thread topics. The orange bar chart shows the interest level of students on the topics $[1, \ldots, K]$. The blue bar chart shows the probability of thread's content to belong to topics $[1, \ldots, K]$

Table 5.1: Notations

Notations	Description
$\boldsymbol{u}(t), \boldsymbol{p}(t)$	Dynamic embedding of student u and thread p at time t
$\boldsymbol{u}(t^-), \boldsymbol{p}(t^-)$	Dynamic embedding of student u and thread p right before time t
$\bar{\boldsymbol{u}}, \bar{\boldsymbol{p}}$	Static embedding of student u and thread p
$\hat{\boldsymbol{u}}(\boldsymbol{t}), \hat{\boldsymbol{p}}(\boldsymbol{t})$	Projected embedding of user u and thread p at time t
$\tilde{\boldsymbol{q}}(\boldsymbol{t})$	Predicted embedding of post at time t
$\theta_t^{u,p}$	Topic distribution of post made at time t
Θ_i	Topic distribution of ith course topic taught in the course.
$\mathcal{P}_u(t)$	Set of threads in which u has posted till time t
$\mathcal{O}$	Set of posts in the training dataset.
$t_{u,p}$	The last time user u posted on thread p.
$\mathbf{w}_t^{u,p}$	The term-frequency vector of the post made by user u on thread p at time t.

learning. Thus, estimating a learner's situational interest not only can provide information about the extent to which a learner is engaged at a given point in time, but it can also present opportunities for interventions designed to sustain such interest, and thus course engagement. learner interest modeling can be used to recommend, for example, the relevant wikipedia articles, forums or research papers a learner should engage with next.

Central to supporting a learner's learning is engagement in effective interactions with the system, while also reducing cognitive load [109]. Indeed, in online learning environments learners expect and experience a high level of customization, interaction with peers, and control during learning [110], all of which result in increased cognitive load. Peer learning specifically,namely learning from each other through discussion, is an important component of the learning process and has a positive impact on learner

learning [49]. However, effectively managing this aspect in the context of an online course can be challenging.

5.2 Student Interest Trajectory for MOOC Thread Recommendation (SITRec)

Problem statement: In the setting of thread recommendation, we are given m students, n threads, and N posts. Each post can be represented as a tuple, $(u, p, t, \mathbf{w}_t^{u,p})$, where $\mathbf{w}_t^{u,p}$ denotes the term-frequency vector of the post made by student u, on thread p at time t. The notations used in this chapter are described in Table 5.1. The problem of thread recommendation can be defined as: For each student, u, find the most relevant threads that she will be interested in. As shown in Figure 5.1 the thread content as well as student interest keeps evolving with time. The interest of the student is further affected by the course topic she is studying. To perform the recommendation task, it is important to predict the future interest of students and properties of threads.
Overview: Our model, SITRec learns embeddings to represent student interests and thread properties. Overall, SITRec comprises of two major operations: update operation and projection operation. The update operation uses two mutually-recursive RNNs to update the embedding of student and thread after the student posts on the thread. To predict student embedding at a future time, the student projection operation leverages the course structure and the elapsed time since last update of student embedding. Lastly, our model also generates a student personalized projected thread embedding which takes into account the idea that a student is more likely to post on the thread she is already associated with. This behavior replicates the notification setting for MOOCs.

5.2.1 Text Representation

The text of each post can be represented as a distribution over few topics because the posts in MOOCs are centred around the topics associated with the course. For this reason, we use topic modeling technique to extract text feature from the post. For extracting the features, we build a dictionary of item vocabularies after filtering the stop words and removing words that occur fewer than 10 times. The content of each post text can be represented as a vector, $\mathbf{w}_t^{u,p} = [w_{t1}^{u,p}, w_{t2}^{u,p}, \ldots, w_{tW}^{u,p}]$, where W is the

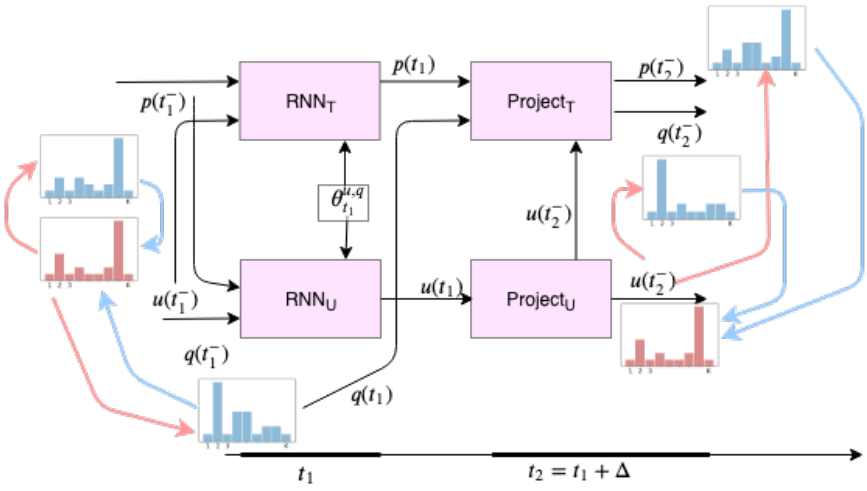


Figure 5.2: The SITRec model: Model illustation for student u (orange) and threads p and q (blue). Student features and thread features influence each other and co-evolve with time. At time t_1, student u posts on thread p, the dynamic embeddings of both u and p are updated with RNN_U and RNN_T, respectively. The projection operation Project_U and Project_T predicts the student and thread embedding , respectively at a future time $(t_1 + \Delta)$.

total number of words in the vocabulary and $w_{tj}^{u,p}$ represents the frequency of word j in the post. The topic distribution vector $\theta_t^{u,p}$ is used to represent the post by student u in thread p at time t and computed using the Latent Dirichlet Allocation (LDA) model [111].

To find the features associated with the course topic taught in ith week of the course, Θ_i, we use LDA to extract the topic distribution of the description of all course materials taught in the ith week. This description of course materials is extracted from the synopsis of course obtained from their respective website [1] .

5.2.2 Embedding layer

We assign each student and thread two embeddings: a static and a dynamic embedding. The static embedding for student $\bar{\boldsymbol{u}} \in \mathbb{R}^m$, encodes the general interest or expertise

[1] https://www.coursera.org/

(which represent the likelihood of student to post on a thread) of students, while that for threads, $\bar{\boldsymbol{p}} \in \mathbb{R}^n$, represents the main topic focussed in the thread. They are obtained using one-hot vectors as inputs, as described in [112]. The dynamic embedding of student, $\boldsymbol{u}(t) \in \mathbb{R}^d$ changes with time and is used to capture the evolving student interest. Similarly, the discussion in threads sometimes deviate as new posts and comments are added. In order to model this dynamic nature, we employ dynamic embedding for each thread, $\boldsymbol{p}(t) \in \mathbb{R}^d$.

5.2.3 Update operation

Whenever a student posts on a thread, both thread embedding and student embedding gets updated. This update is modeled by two mutually-recursive Recurrent Neural Networks (RNNs). The hidden states of the RNN_U and the RNN_T represent the student and thread embeddings, respectively. The two RNNs are coupled together because thread embedding affects the student embedding and student embedding affects that of thread. As shown in Figure 5.2, when student u posts on thread p, RNN_U updates the embedding $\boldsymbol{u}(t)$ by using the embedding $\boldsymbol{p}(t^-)$ of thread p right before time t and text representation of the post $\theta_t^{u,p}$ as inputs. Similarly, RNN_T updates embedding $\boldsymbol{p}(t)$ by using the embedding $\boldsymbol{u}(t^-)$ of student u right before time t and text representation of the post $\theta_t^{u,p}$ as inputs. More formally,

$$\boldsymbol{u}(t) = \sigma(\boldsymbol{W}^u[\boldsymbol{u}(t^-), \boldsymbol{p}(t^-), \theta_t^{u,p}, \Delta_u]), \tag{5.1}$$

$$\boldsymbol{p}(t) = \sigma(\boldsymbol{W}^p[\boldsymbol{p}(t^-), \boldsymbol{u}(t^-), \theta_t^{u,p}, \Delta_p]), \tag{5.2}$$

where Δ_u denotes the time since u's previous post on any thread and Δ_p is the time since last post on thread p, $\theta_t^{u,p}$ is the text feature vector of the post. The matrices $\boldsymbol{W}^u, \boldsymbol{W}^p \in \mathbb{R}^{(2d+F+1)\times d}$ are the parameters of RNN and F is the number of features associated with the post.

5.2.4 Projection Operation

The projection operation predicts the future trajectory of student interests based on course structure and student personalized embeddings of threads based on the nature of new posts made on the thread.

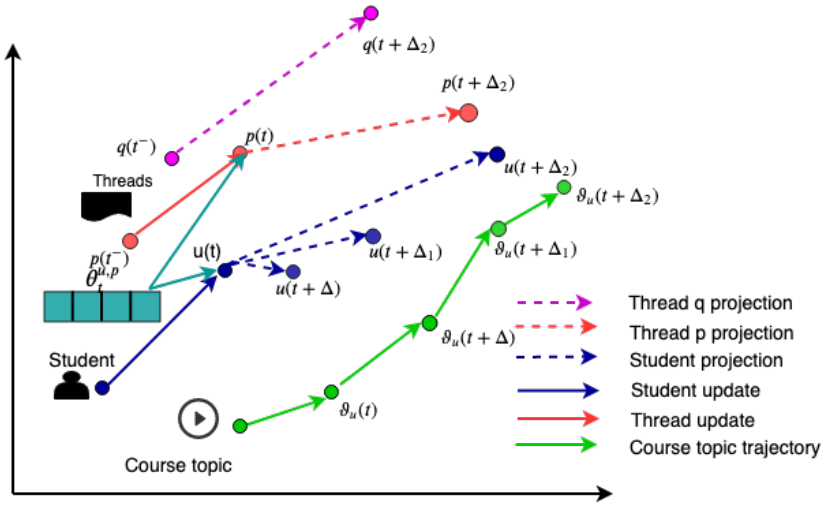


Figure 5.3: Projection Operation : This figure shows the key idea behind projection operation. At time t, student u posts in thread p with post features $\theta_t^{u,p}$. The projected embedding of student u is shown for different elapsed times $\Delta < \Delta_1 < \Delta_2$. The course topics ϑ_u represents the topics u is studying at different times. The embeddings of the two threads, p and q are also shown. After elapsed time Δ_2 thread p's embedding is projected closer to u's embedding while thread q on which u did not post in the past is projected farther from u's embedding.

Student Projection

In this section, we will describe how we obtain the future embedding trajectory of a student. The motivation behind the student projection operation is two-folds: 1) as time elapses student interest drifts farther from the original, 2) the course topic a student is interested in is an important factor in deciding her future interest. As shown in Figure 5.3, a student u posts at time t and the RNN layer outputs her interest embedding $\boldsymbol{u}(t)$. After a short duration Δ_1 since t, the student's projected embedding $\boldsymbol{u}(t+\Delta_1)$ is close to her previously observed embedding $\boldsymbol{u}(t)$. As more time $\Delta_2 > \Delta_1 > \Delta$ elapses, the projected embedding drifts farther from $\boldsymbol{u}(t)$ and the course topic embedding ,$\vartheta_u(t)$, helps in guiding the evolution of projected embedding of the student.

The first step in projecting a student embedding is to determine the topic, she is interested in, which is determined as, $\vartheta_u(t) = i|\Theta_i = argmin_\Theta||\Theta - \theta_t^{u,p}||_2$, where Θ_i

is the topic distribution of ith week course content and $\theta_t^{u,p}$ is the topic distribution of post made by student u on thread p at time t. Then, to predict the projected student embedding, we incorporate the context features: current course topic embedding, $\vartheta_u(t)$ and the elapsed time since last update, Δ along with student current embedding $\boldsymbol{u}(t)$ as input. Since simply concatenating the context features and passing through linear layer has proved to be ineffective in modeling the interaction between the concatenated input features, we follow the procedure suggested in Latent Cross [113]. We describe how we obtain the feature-context vector below.

To incorporate the context feature f, we first convert f to a feature-context vector $\boldsymbol{w}_f \in R^d$ using a linear layer $\boldsymbol{w}_f = \boldsymbol{W}_f f$. The weights of the linear layer, $\boldsymbol{W}_f$ is initialized by a 0-mean Gaussian. We represent the time-context vector as $\boldsymbol{w}_\Delta$ and the course topic-context vector as $\boldsymbol{w}_\vartheta$. The projected embedding is then obtained as element-wise product of the context vector and the previous embedding as,

$$\hat{\boldsymbol{u}}(t+\Delta) = (\mathbf{1} + \boldsymbol{w}_\Delta + \boldsymbol{w}_\vartheta) * \boldsymbol{u}(t) \tag{5.3}$$

Thread Projection

Thread projection layer projects thread embedding personalized to each student based on the nature of posts made on the thread. It is essentially important to capture the temporal dynamics of threads. Intuitively, a student is likely to be interested in a thread if another student posts on the thread which she is already associated with. The level of interest further increases if another student comments on the student's post. This also reflects the notification setting for discussion forum, where student gets notified whenever any posts/comments are made on threads that the student has interacted with. Motivated by this, we develop a thread projection layer which learns a student-personalized thread embedding such that the thread embedding is projected closer to student embedding based on nature of posts/comments made on the thread after the student's last interaction.

The projected thread embedding with respect to student u is obtained as,

$$\hat{\boldsymbol{p}}_u(t+\Delta) = \frac{\zeta_{u,p}(t+\Delta)}{1+\zeta_{u,p}(t+\Delta)}\boldsymbol{u}(t) + \frac{1}{1+\zeta_{u,p}(t+\Delta)}\boldsymbol{p}(t), \tag{5.4}$$

where ζ-factor, $\zeta_{u,p}(t+\Delta)$ defines how much closer the projected thread embedding is to the student embedding. The higher the value of ζ-factor, the closer is the projected

thread embedding to the student embedding. Naturally ζ-factor should have different terms for posts on the thread and comments on student's post as they induce different level of excitement among students [48]. This excitement also fades as the time elapses owing to the ageing of the threads. As a result we define ζ-factor as:

$$\begin{aligned}\zeta_{u,p}(t+\Delta) = \mathbf{1}_{\mathcal{P}_u}\Big(& \sum_{t_{u,p}<t_p<t+\Delta} e^{-\alpha(t_p - t_{u,p})} \\ & + \sum_{t_{u,p}<t_r<t+\Delta} e^{-\beta(t_r - t_{u,p})}\Big),\end{aligned} \tag{5.5}$$

where $\mathbf{1}_{\mathcal{P}_u}$ is 1 if u posted in p , otherwise 0, $t_{u,p}$ is the last time user u posted on p, α and β are the scalar weights given to the excitement level induced by a new post and replies on the student's posts on the thread p, t_p and t_r are the timestamps of posts made on the thread p and the timestamps of the explicit replies made on the student's post on p, respectively.

5.2.5 Recommendation

Similar to JODIE model [42], we predict the embedding of the next thread that will interest the student. We make this prediction using the projected student embedding $\hat{\boldsymbol{u}}(t+\Delta)$ and the embedding of thread $\boldsymbol{p}(t)$ of thread p (the thread on which u last posted on). The reason we include $\boldsymbol{p}(t)$ is that students often interact with the same item consecutively and including the item embedding helps to ease the prediction. The prediction is made using a linear layer as follows:

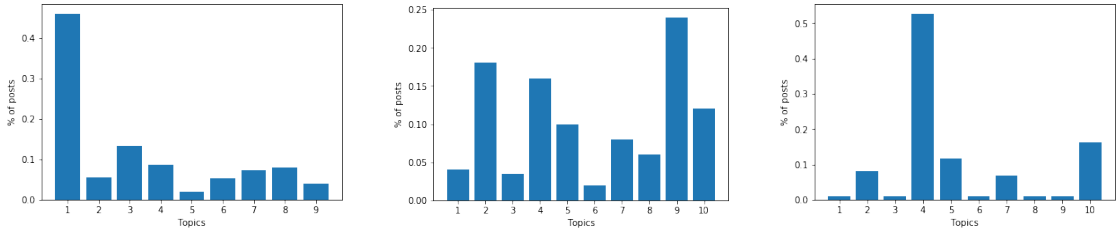
$$\tilde{\boldsymbol{q}}(t+\Delta) = \boldsymbol{W}[\hat{\boldsymbol{u}}(t+\Delta), \bar{\boldsymbol{u}}, \boldsymbol{p}(t), \bar{\boldsymbol{p}}] + \boldsymbol{B}, \tag{5.6}$$

where $\boldsymbol{W} \in \mathbb{R}^{(m+n+2d)\times(n+d)}$ is the weight matrix and $\boldsymbol{B} \in \mathbb{R}^{n+d}$ is the bias vector in the linear layer.

Having generated the predicted thread embedding at time $t+\Delta$, we find the candidate threads for recommendation using nearest-neighbor search which are closest to the predicted thread embedding.

5.2.6 Network Training

We train our model to minimize the Euclidean distance between the predicted thread embedding and the ground truth thread embedding everytime a student posts on a



Algo ML comp

Figure 5.4: Dataset statistics in terms of posts per topic.

Table 5.2: Dataset Statisitics

Dataset	Threads	Posts	Learners	Weeks
ml	5310	40050	6004	15
algo	1323	9274	1833	9
comp	4860	17562	3060	14

thread. We calculate the total loss as,

$$Loss = \sum_{u,p,t,\boldsymbol{w}_t^{u,i} \in \mathcal{O}} ||\tilde{\boldsymbol{q}}(t) - [\bar{\boldsymbol{p}}, \hat{\boldsymbol{p}}_u(t)]||_2 + \lambda_U ||\boldsymbol{u}(t) - \boldsymbol{u}(t^-)||_2 + \lambda_T ||\boldsymbol{p}(t) - \boldsymbol{p}(t^-)||_2, \quad (5.7)$$

where $\mathcal{O}$ is set of posts in training sample, λ_U and λ_T are regularization parameters for temporal smoothness of student and thread embeddings, respectively. The complete parameter space in our training models is $\Omega_{\text{update}} = \{\boldsymbol{W}^u, \boldsymbol{W}^p\}, \Omega_{\text{project}} = \{\boldsymbol{W}_\Delta, \boldsymbol{W}_\vartheta, \alpha, \beta\}, \Omega_{\text{Rec}} = \{\boldsymbol{W}, \boldsymbol{B}\}, \Omega_{reg} = \{\lambda_U, \lambda_T\}$.

5.3 Experimental Settings

To comprehensively evaluate the performance of our proposed SITRec model, we design different strategies to evaluate the effectiveness of the model.

5.3.1 Dataset

We use three real-world datasets to evaluate the performance of our model. These datasets are obtained from Coursera course offering for three courses, namely, Machine

Learning (ml), Algorithms, Part I (algo), and English Composition I (comp), in 2012. Table 5.2 gives details on these datasets. These datasets, in addition to varying in the size of users and density of interaction, also comprises of different user behavior in terms of posts per topic. As shown in Figure 5.4, ml has the most diversified posts, pertaining to different topics, while algo and comp have most posts related to one topic.

5.3.2 Comparison Approaches

We compare our model with the following approaches:

- **Popularity-based (POP)**: This is a simple baseline that ranks threads from most to least popular according to their popularity.
- **Recency-based (REC)**: This is also a simple method that ranks threads from oldest to newest based on the time the most recent post was made on the thread.
- **Personalized Recency-based (USER-REC)**: This method ranks threads from oldest to the newest based on the time the user interacted with the thread.
- **Adaptive Matrix Factorization (AMF)** [**46**]: This is an Adaptive Matrix Factorization based method which finds similar users and recommends those threads to a user which similar users have posted on.
- **Point Process based (PPS)** [**48**]: This is a Point Process based method which calculates the probability that a user will post on a thread. It uses a heuristic that a post on a thread and an explicit reply on a user's post increases the likelihood of participation of the user on the thread in different manner.
- **Deep Coevolutionary (DeepCo-evolve)** [**41**]: A co-evolutionary model that updates user and item embeddings when a user interacts with an item using RNN. To predict whether user will interact with item it employs point process technique where the probability of the interaction decays with time.
- **JODIE** [**42**]: JODIE is state-of-the-art model for predicting a user's interaction with item. It is also co-evolutionary model that projects user embedding using temporal attention layer after some elapsed time Δ since user's previous interaction.

Table 5.3: Performance comparison on three datasets for all methods in terms of Mean Average Precision (MAP) @5. The best and the second best results are highlighted by **boldface** and underlined respectively. Gain% denotes the performance improvement of SITRec over the best baseline.

Methods	**Algo**	**ML**	**Comp**
POP	0.102	0.005	0.001
REC	0.020	0.090	0.066
USER-REC	0.338	0.150	0.221
AMF [46]	0.091	0.005	0.253
PPS [48]	0.362	0.152	0.332
DeepCo-evolve [41]	0.112	0.088	0.162
JODIE [42]	0.397	0.253	0.212
SITRec	**0.561**	**0.400**	**0.393**
Gain%	41.310	58.102	18.373

Metrics

We evaluate forum recommendation using the standard ranking metric Mean Average Precision ($MAP@N$). In our experiments, we set $N = 5$.

$$AP_u@N = \frac{\sum_{n=1}^{N} P_u@n \times post(n)}{min|R_u, N|}, \tag{5.8}$$

where R_u is the set of threads student u posted on during the test time interval and $post(n)$ is a binary function that describes whether the user has posted in the nth thread. $P_u@n$ denotes the precision at n. Finally, MAP is obtained by averaging the AP values of all the users.

5.3.3 Evaluation Methodology

Model Training and Parameter Selection

We perform a series of pre-processing on the text of posts. For preparing the feature associated with each post we process the text by i) removing url links, punctuations and

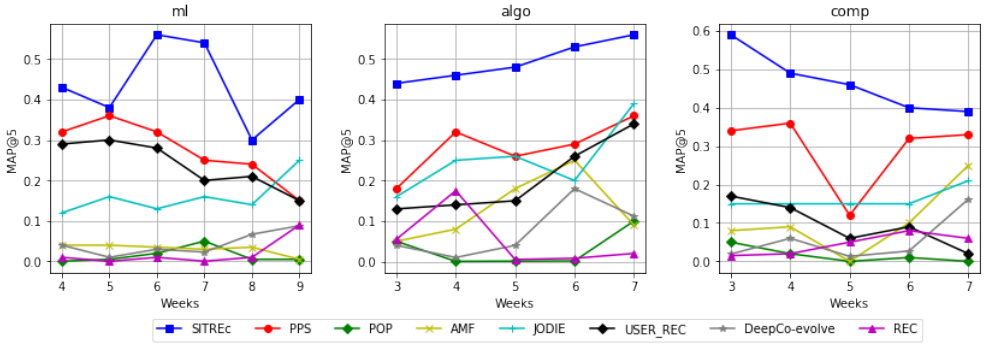


Figure 5.5: Plot of recommendation performance over different lengths of the training time window $T1$ on all datasets. Our model, SITRec significantly outperforms every baseline.

words that contain digits, ii) convert all words to their respective base forms, iii) remove stopwords and (iv) remove words that appear fewer than 10 times. Then, we obtained a bag-of-words representation of each text. The process used for obtaining features associated with each post from bag-of-words representation is explained in Section 3.1. The number of topics used in LDA algorithm is same as the number of topics in the course as extracted from the course syllabus because we assume that forums are centered around the topics of course content. We also run LDA on the course syllabus obtained from the course website.

For all the datasets, we tried the embedding dimensions from $[5, 10, 15, 20, 25]$ and chose the value that gave the best performance. The values of α and β required in thread projection were selected from $[0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0]$. We found that $\alpha = 0.5$ and $\beta = 0.001$ gives the best performance for algo dataset, $\alpha = 0.5$, $\beta = 0.005$ gives the best performance for ml dataset and α =0.5, $\beta = 0.1$ gives the best performance for comp dataset. We used learning rate of 0.001 and t-batch algorithm [42] for creating the batches in our experiments.

5.4 Results and Discussion

5.4.1 Performance Evaluation

Table 5.3 shows the the recommendation performance of our model and the baselines over for all the dataset when the training time T_1 is set to $W-1$ week, where W is the duration in which forums are active and testing time interval $(T_2 - T_1)$ is one day after. The value of W is $10, 8, 8$ for ml, algo, and comp, respectively. Since learners drop out from the course with time leading to reduction in forum activities, these values of W is less than those mentioned in Table 5.2.

As seen in the table 5.3, our proposed SITRec significantly outperforms existing methods in all the datasets. Among the simple baselines (POP, REC, USER-REC), USER-REC performs better than the rest. This confirms that users tend to post comments on threads they are already associated with. USER-REC performs better than AMF because AMF does not take into account the posts that the user has already posted on. Since repetitive behavior of users is an important signal for making prediction of next thread and AMF fails to take that into consideration, it is outperformed by USER-REC and PPS methods. Among the co-evolutionary models proposed in the literature, JODIE significantly outperforms Deep Co-evolve which is in agreement to [42]. Since JODIE takes into consideration the last thread on which user posted to predict the embedding of the next thread, it performs better than DeepCo-evolve Finally, SITRec outperforms all the baselines. There is no clear winner among the baselines: JODIE performs better than PPS on algo and ml dataset while PPS performs better than JODIE on comp dataset. This could be because in comp being English Composition dataset, the discussions in each thread is longer, leading to more activity notifications and students tending to reply on same thread, while in engineering courses like ml and algo learners are expected directly answer each other's questions than holding long discussion [48].

The fact that SITRec outperforms the JODIE baseline confirms both our hypothesis regarding MOOC forums. First, it is important to consider how the user's interest evolves (by taking into account the course topic that the student is studying). Second, user's interest in a thread increases if she has already posted in that thread and if someone replies on his post. The fact that SITRec outperforms other baselines which

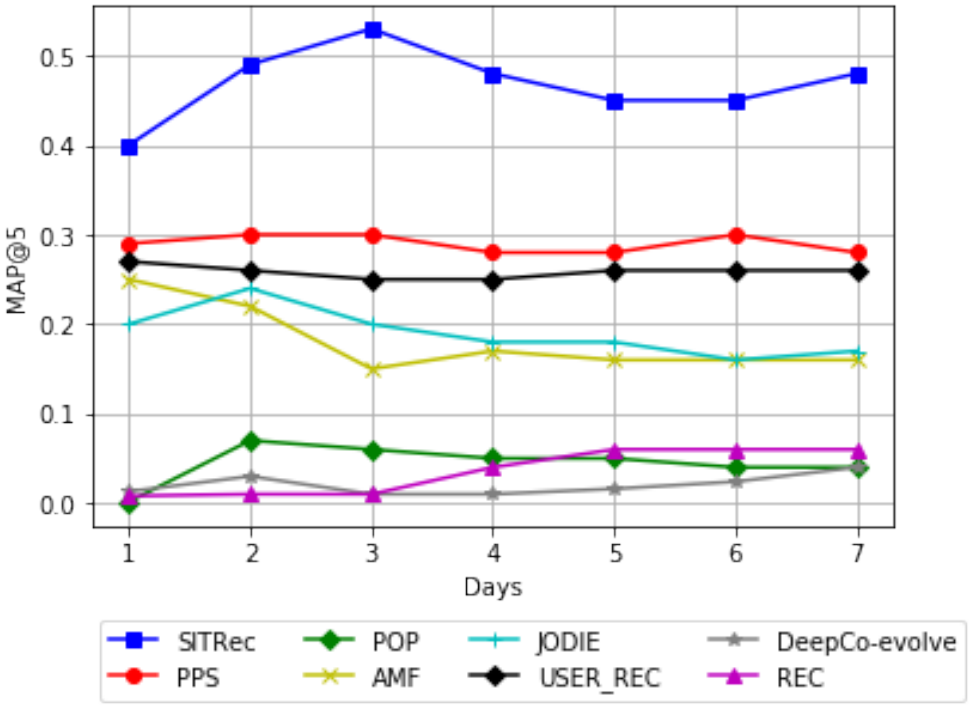


Figure 5.6: Recommendation performance for algo dataset by varying testing window length.

do not consider the evolving nature of user interest and thread's properties emphasizes the benefit of the co-evolutionary RNNs in capturing the dynamic nature involved in thread activities.

Robustness towards proportion of data

In this experiment, we validate the robustness of SITRec by varying the data taken as training set and test set and comparing the performance of the algorithm with other baseline methods. In the first setting, we hold the testing interval fixed to one day and vary the training data size from 1 week to $W-1$ weeks, where W is the duration in which forums are active and testing time is one day after. Figure 5.5 shows the performance of the various methods in this setting. Overall, we see that our model significantly outperforms the baselines in each case, achieving 7% to 190% improvement over the

Table 5.4: Comparing variants of the proposed model. Best results are indicated in **bold**

Dataset	**algo**	**ml**	**comp**
SITRec-Student Projection	0.53	0.37	0.30
SITRec-Thread Projection	0.35	0.29	0.27
SITRec-Text Features	0.46	0.34	0.27
SITRec	**0.56**	**0.40**	**0.39**

strongest baseline. Another interesting observation is that even when the training data is of small interval, (i.e., when training data consisted of only few weeks) SITRec gives good performance compared to other models.

In the second setting, we hold the length of the training interval fixed at $W-2$ weeks to allow sufficient number of posts in the test week and vary the length of the testing interval from 1 day to 7 days. Figure 5.6 shows the recommendation performance over different lengths of the testing time window ΔT for the algo dataset. Our model, SITRec outperforms the baseline methods for all the values of ΔT. Even when the length of test set interval increases the performance of our model does not degrade, in fact improves in some cases. This can be explained by the intuition that every action taken by a student improves the learnt embedding of student interest. Thus, our model is robust towards the length of testing interval and is able to model student behavior over long period of time as well. Since the performance on other datasets was similar, we omitted the chart of other datasets.

5.4.2 Ablation Study

In order to verify the effectiveness of the modification we introduced in this work, we run an ablation study to check the importance of each individual component. The results are provided in Table 5.4. The variants of our models are:

- **SITRec-Student Projection**: In this variant, we do not predict future student embedding and the embeddings are only updated when student makes a post on a thread.

- **SITRec-Thread Projection**: In this variant of SITRec model, we do not project a thread embedding specific to each student. We use the embedding of thread obtained right after the update operation.

- **SITRec-Text Features**: In this variant of SITRec, we remove the text feature input to RNN_U and RNN_T models.

The results are obtained by taking $W - 1$ weeks as training interval and 1 day as testing interval for each dataset. Removal of the student projection operation is shown to reduce the performance of model to some extent, however, removal of thread projection causes a drastic reduction in performance of the model. This suggests that on MOOC forums students tend to post on the threads they have already visited before. This factor plays an important role in deciding the thread to recommend when multiple threads on same topic exist. Without thread projection layer, even if SITRec predicts correct topic of interest for student, it fails to identify particular thread to recommend. As a result, identifying the thread after determining the topic of interest is easier for ml dataset compared to algo and comp. At last, to investigate the effectiveness of textual features of posts and comments in a thread, SITRec-Text feature is introduced where no textual features are fed to the two RNNs in the update operation. We find decrease in performance of the model suggesting that textual features help in enhancing the model performance.

5.5 Thread Recommendation on Generalized Platforms

We also develop models to improve the recommendations of threads on generalized. These platforms such as, Reddit, Wikipedia, and StackOverflow provide an open and broad subject for people to discuss their ideas and express their thoughts. In the light of dynamic user interest and evolving item properties, we develop representation learning techniques to learn dynamic embeddings of users and items. However, learning embeddings on these platforms is a challenging task because the user interest keep evolving. This evolution can be captured from 1) interaction between user and item, 2) influence from other users in the community. The existing dynamic embedding models only consider either of the factors to update user embeddings. However, at a given time,

user interest evolves due to a combination of the two factors. To this end, we propose Influence-aware and Attention-based Co-evolutionary Network (IACN) [11].

The motivation is that when a user interacts with an item her interest at that time can be determined from the interaction features. However, as time elapses, the interest of the user drifts and tends to be more driven by the influence of other users. The key components in the IACN model are:

Interaction modeling layer: The interaction modeling layer is responsible for updating the embedding of corresponding users and items when they interact with each other. We leverage the attention mechanism to identify which interactions are important for determining the updated embedding of entities (users and items) involved in the interaction. As shown in Figure 5.7, when a user interacts with an item, $\mathrm{ATT_U}$ updates the embedding of the user by adaptively assigning weights to its previous interactions. Similarly, $\mathrm{ATT_I}$ updates the embedding of the item based on its past interaction.

Influence modeling layer: We design a "relation revealing" attention-based operation to capture the relation between users and then update the embedding of a user when any user who influences the user interacts with an item. As shown in Figure 5.7, when a user interacts with an item, it triggers a drift of interest of other users towards the item.

Fusion layer: To learn future embedding of a user, we design a novel fusion layer that integrates the embedding from interaction and influence modeling layer. When an interaction occurs, the user embedding is determined solely by the interaction modeling layer because user interaction reveals the user's current interest [114]. As time progresses user embedding drifts further apart from the interaction embeddings. As shown in Figure 5.7, the future user embedding is computed by additively combining the influence-based embedding and the interaction-based embedding where the contribution of the interaction model decays while that of the influence model increases with time.

To recommend the next item which the user will interact with, IACN predicts an embedding for the next item and uses Locality Sensitive Hashing [115] to find the item whose embedding is most similar to the predicted item embedding. Summary of this work's major contributions are:

- We study the contribution of both the interaction model and the influence model

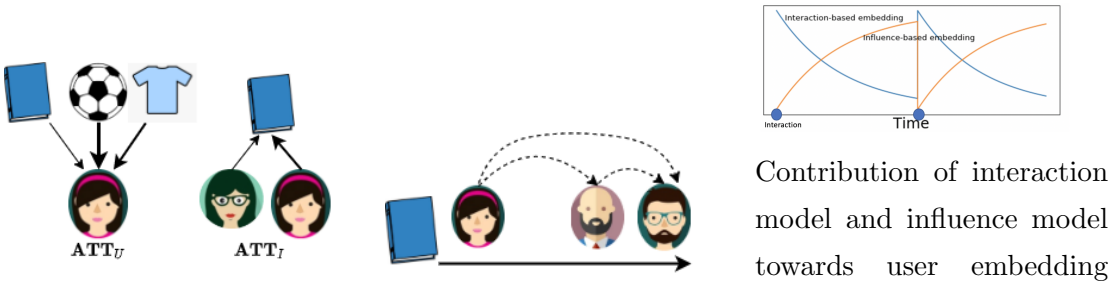


Figure 5.7: A simplified diagram showing the main components of IACN.

in predicting embeddings for the recommendation.

- We design a co-evolutionary network using two attention layers to update the embeddings of users and items. The attention layers help in improving the performance of our model along with providing insight into different user behaviors.
- We introduce a novel method to model the influence of other users on a user and integrate it with the interaction model to obtain the user embedding at query time
- We conduct experimentation on the real-world dataset and demonstrate the superiority of our model over state-of-the-art baselines over various domains.

5.6 Notations, Definitions, and Preliminaries

Notations. Given m users and n items, we denote the temporal list of N observed interactions as $\mathcal{O} = \{o_j = (u_j, i_j, t_j, \boldsymbol{q}_j) \forall j \in N\}$, where $u_j \in \{1, \ldots, m\}, i_j \in \{1, \ldots, n\}, t_j \in \mathbb{R}^+$ and $\boldsymbol{q}_j \in \mathbb{R}^F$ represent the interaction features. For simplicity, we define $\mathcal{O}^u = \{o_j^u = (i_j, t_j, \boldsymbol{q}_j)\}$ as the ordered listed of all interactions related to user u, and $\mathcal{O}^i = \{o_j^i = (u_j, t_j, \boldsymbol{q}_j)\}$ as the ordered list of all interactions related to item i.

These interactions result in formation of a network where nodes represent either a user or an item. As users interact with an item, an edge is created between them. Due to the sequential nature of these interactions, this network keeps evolving with time. We can formally define the temporal interaction network as,

Definition 5.6.1. Temporal Interaction Network Temporal interaction network is a bipartite graph with edges annotated by chronological interactive events between nodes (users and items) and is denoted as $G =< V, E; \mathcal{O} >$, where $V = \{v_1, v_2, \ldots, v_{|V|}\} = U \cup I$, E denotes the set of edges. Each edge $(u, i) \in E$ between user u and item i is annotated by chronological interactions between user u and item i.

To learn embeddings of users and item in temporal interaction network, Co-evolutionary models have been studied in [114, 41, 116]. We now formally define Co-evolutionary model as:

Definition 5.6.2. Co-evolutionary models Co-evolutionary models consist of two interwoven and interdependent layers, such that the output of one effects the output of the other and vice-versa.

In addition, users are influenced by other users in the social network. Point process models the discrete sequential events by assuming that historical events before time t can influence the occurrence of the current event [117]. Their application in modeling latent influence between users in a social community has been studied in literature [118, 119, 120, 121]. Here we describe a general outline of how point process is used in modeling the social influence.

Point processes are characterized using conditional intensity function $\lambda(t)$. The conditional intensity associated with a temporal point process is defined to be the expected infinitesimal rate at which events are expected to occur around time t given the history, $\mathcal{H}(t)$. The conditional probability of observing an event in a small time window $[t, t+dt)$ is $\lambda(t)dt$. Algebraically, $\lambda(t)dt = \mathbb{P}\{\text{event in}[t, t+dt)|\mathcal{H}(t)\} = \mathbb{E}[dN(t)|\mathcal{H}(t)]$. Following this idea, social influence methods [121, 118] attempt to model the rate that user u adopts item i at time $t + \Delta$ influenced by the interaction of user v with item i at time t as,

$$\lambda_{i,t,v}(t + \Delta, u) = \alpha_v \theta_{u,v} exp(-\Delta)$$

where α_v represents the influence of user v on other users, $\theta_{u,v}$ is the strength of relation from user v to u and $exp(-\Delta)$ models the decay of influence over time. When we arrange different user's interaction with items as a sequence according to ascending time, we can find which users influence other users to interact with the item. These users form local user neighborhood for the user in consideration. As the user interacts with more items,

its neighborhood keeps evolving. We can formally define the local user neighborhood of a user as follows:

Definition 5.6.3. Local user neighborhood Given a temporal interaction network $G =< V, E; \mathcal{O} >$ representing the observed user-item interactions, the local user neighborhood, $\mathcal{N}_u(t)$ of a user u are all those users $v \in U$ which are associated with at least one item before u interacted with it. Mathematically, when an interaction $o_j = (u_j, i_j, t_j, \boldsymbol{q}_j)$ is observed the local user neighborhood is updated as, $\mathcal{N}_u(t_j) = \mathcal{N}_u(t_j^-) \cup \mathcal{U}^i(t_j^-)$, where $\mathcal{U}^i(t_j^-)$ is the set of user who interacted with item i before time t_j and $\mathcal{N}_u(t_j^-)$ is the local neighborhood of user u right before time t_j.

5.6.1 Model Architecture

We will now describe each layer in IACN in detail.

Embedding layer. We assign each user and item two embeddings: a static and a dynamic embedding. The static embedding encodes the long-term stationary properties while the dynamic embedding encodes the dynamic properties. This decision is made by following the setting in [114] such that static embeddings, for a user, u, $\bar{\boldsymbol{u}} \in \mathbb{R}^m$ and item i, $\bar{\boldsymbol{i}} \in \mathbb{R}^n$ represent the long-term properties of the entities. While dynamic embeddings $\boldsymbol{u}(t) \in \mathbb{R}^d$ and $\boldsymbol{i}(t) \in \mathbb{R}^d$ at time t, respectively model the time-varying behavior and features.

Interaction modeling layer. The interaction modeling layer updates the embedding of a user and an item when the user interacts with the item. In particular, when an interaction $o_j = (u_j, i_j, t_j, \mathbf{q}_j)$ is observed, the dynamic embedding of the involved user u and item i is updated. For simplicity of notations we drop the j subscript in the following section to represent static embeddings as $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{i}}$ and dynamic embedding as $\boldsymbol{u}(t)$ and $\boldsymbol{i}(t)$.

To obtain interaction-based embedding of u and i, we consider their past interactions till time t $\mathcal{O}^u(t) = \{o_1^u, o_2^u, \dots o_p^u\}$ such that $t_p \leq t$ and $\mathcal{O}^i(t) = \{o_1^i, o_2^i, \dots o_q^i\}$ such that $t_q \leq t$, respectively. We use attention mechanism to compute the *importance* of past interactions in determining the updated embedding of u as:

$$e_k^u(t) = a(\boldsymbol{W}^i \boldsymbol{i}_k(t_k^-), \boldsymbol{W}^u \boldsymbol{u}(t^-)) + a(\boldsymbol{W}^q \boldsymbol{q}_k, \boldsymbol{W}^u \boldsymbol{u}(t^-)) \tag{5.9}$$

where $\boldsymbol{i}_k(t_k)$ represents the dynamic embedding of item occurring at kth interaction in $\mathcal{O}^u(t)$, t^- represents the time right before the time t, $\boldsymbol{W}^u, \boldsymbol{W}^i \in \mathbb{R}^{d\times d}$, $\boldsymbol{W}^q \in \mathbb{R}^{d\times F}$ are the weight matrices and d and F are the embedding size and the number of features associated with an interaction, respectively. The intuition is as follows, the first term computes importance of i's features at the time of interaction to predict u's future embedding. The second term introduces the level of contribution the interaction features have towards the evolution of u. In our experiments, we used a as the dot product between the two vectors.

Having computed the attention coefficients, $e^u(t)$, corresponding to all historical interactions involving u, we compute the new embedding of u as:

$$\boldsymbol{u}(t) = \sigma\Bigg(\sum_{j,o_j\in\mathcal{O}^u(t)} \alpha_j^u(t)\boldsymbol{W}^i\boldsymbol{i}_j(t_j)\Bigg), \alpha_j^u(t) = \frac{exp(e_j^u(t))}{\sum_{k,o_k\in\mathcal{O}^u(t)} exp(e_k^u(t))}, \tag{5.10}$$

where σ is introduced for non-linearity. Here we have described an attention layer to update the embedding of user u. To update the embedding of i, we employ the same two operations with interactions associated with the item.

Influence modeling layer One of the major novelty of our method is that we introduce a time-varying self-attention based influence model for predicting user's future interest. The idea is to leverage the knowledge of evolution of a user's neighbors to predict future embedding of the user. Modeling neighborhood influence in temporal interaction network poses specific challenge as the influence of an interaction on a user is driven by both the relation between users and time elapsed since the interaction.

Our model captures the influence of u's local neighborhood on u's embedding by modeling a function that outputs a representation vector, influence embedding, $\boldsymbol{I}_u(t)$. This influence embedding is governed by an aggregation function parameterized by the temporal interaction sequence involving user neighborhood. Influence-based embedding at time $t+\Delta$ is computed as:

$$\boldsymbol{I}_u(t+\Delta) = \sum_{v\in\mathcal{N}_u(t)t<t_v<t+\Delta} \theta_{v,u} exp(-\delta_u(t+\Delta-t_v))\boldsymbol{v}(t_v), \tag{5.11}$$

where $\theta_{v,u}$ models the influence user v has on u and $exp(-\delta_u(t+\Delta-t_v))$ models decay of the influence over time with user-specific parameter δ_u and $\mathcal{N}_u(t)$ is the local user neighborhood of u. To model the level of influence a user v has on the other u, we again

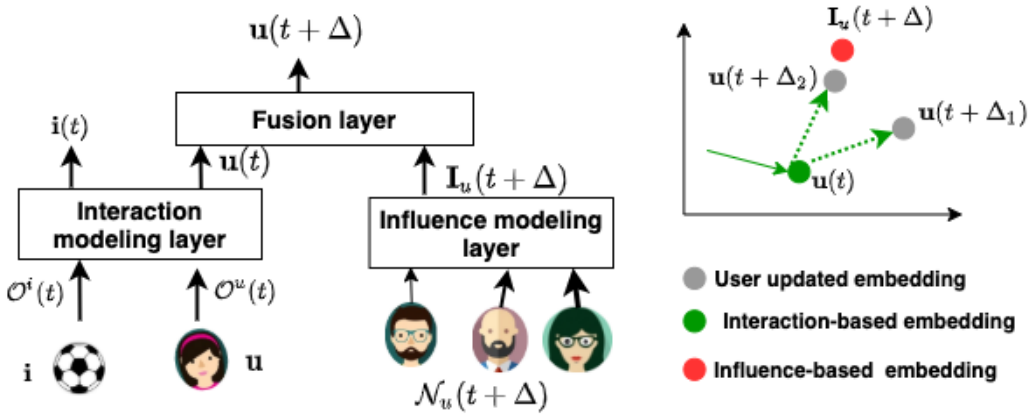


Figure 5.8: The IACN model: After an interaction $(u, i, t, \mathbf{q})$, the dynamic embeddings of u and i are updated in the Interaction modeling layer. The Influence modeling layer predicts the user embedding at time $t+\Delta$, $\mathbf{u}(t+\Delta)$ by taking influence vector $\mathbf{I}_u(t+\Delta)$ into consideration. The figure on the right side shows how influence modeling layer updates user embedding. As more time elapses, $(\Delta_2 > \Delta_1)$, the user embedding tends to be closer to $\mathbf{I}_u(t)$.

utilize the attention mechanism, i.e.,

$$\theta_{v,u} = \begin{cases} a(\boldsymbol{W}_1^l \boldsymbol{v}(t_v), \boldsymbol{W}_2^l \boldsymbol{u}(t)) & \text{if } v \in \mathcal{N}_u(t), \\ 0 & \text{otherwise} \end{cases}, \tag{5.12}$$

where $\boldsymbol{W}_1^l$ and $\boldsymbol{W}_2^l$ are the weight parameters of the attention mechanism. Due to peer engagement and affinity between users, θ is sparse as users tend to indulge in discussions with users of their community. For validating this, we computed the average length of local user neighborhood in 'Wikipedia' dataset (described in section 5.1). We find that with 8227 users, the number of non-zero values in θ is 191,307. The average length of local user neighborhood is only 23.2.

Fusion layer To integrate the signals from interaction layer and influence layer, we introduce a fusion layer. This layer predicts embeddings of user at time t by taking into account the user embedding, the influence embedding, and the time elapsed since u's last interaction, Δ. The motivation behind constructing this layer is that a user interest keeps evolving even when it is not interacting with any item and as more time

elapses the future embedding is farther from the user embedding. Furthermore, the interactions from the user local neighborhood influences the user interest which becomes more pronounced as more time elapses. To model this, we employ a kernel function such that the user embedding $\boldsymbol{u}(t+\Delta)$ will continue to deterministically decay (at different rates for different users) from interaction-based embedding $\boldsymbol{u}(t)$ towards influence-based embedding $\boldsymbol{I}_u(t+\Delta)$. Thus, we extrapolate a user embedding at a future time as:

$$\boldsymbol{u}(t+\Delta) = \boldsymbol{u}(t) + (\boldsymbol{I}_u(t+\Delta) - \boldsymbol{u}(t))(1 - exp(-\beta_u \Delta)), \tag{5.13}$$

where β_u is a parameter learned while training the model. On the interval $[t, t+\Delta)$, the u's embedding follows an exponential curve that begins at $\boldsymbol{u}(t)$, when $\Delta \to 0$ and decays towards $\boldsymbol{I}_u(t)$ (as $t \to \infty$, if extrapolated).

Recommendation layer. Once we predict users' embeddings at time $t+\Delta$, we predict the embedding of the next item. For this we use the updated user embedding $\boldsymbol{u}(t+\Delta)$ and the embedding of item that u last interacted with at time t, $\boldsymbol{i}(t)$. The predicted item embedding is:

$$\hat{\boldsymbol{i}}(t+\Delta) = \boldsymbol{W}[\boldsymbol{u}(t+\Delta), \bar{\boldsymbol{u}}, \boldsymbol{i}(t), \bar{\boldsymbol{i}}] + \boldsymbol{B}, \tag{5.14}$$

where $\boldsymbol{W}$ is the weight matrix and $\boldsymbol{B}$ bias vector which make the linear layer. Then we recommend the items with the closest embedding with the predicted embedding. This step can be done in near-constant time by using LSH [115].

5.6.2 Network training

We train our model to minimize the Euclidean distance between the predicted item embedding and the actual item embedding everytime a user interacts with an item. We calculate the total loss as,

$$\mathcal{L} = \sum_{(u,i,t,\boldsymbol{q}) \in \mathcal{O}} ||\hat{\boldsymbol{i}}(t) - [\bar{\boldsymbol{i}}, \boldsymbol{i}(t)]||_2 + \lambda_U ||\boldsymbol{u}(t) - \boldsymbol{u}(t^-)||_2 + \lambda_I ||\boldsymbol{i}(t) - \boldsymbol{i}(t^-)||_2,$$

where λ_U and λ_I are regularization parameters for temporal smoothness of user and item embeddings, respectively.

5.7 Experimental Settings

To comprehensively evaluate the performance of our proposed IACN model, we design different strategies to evaluate the effectiveness of the model. Our experiments are designed to answer the following research questions:

1. **RQ1:** How does IACN perform compared with other state-of-the-art recommendation models?

2. **RQ2:** What is the influence of various components in the IACN architecture?

Datasets. We used 4 public datasets and followed the same preprocessing steps as used in [114]. Thus,we selected 1000 most active items in each dataset.

- **Wikipedia dataset:** Public dataset consisting of one month of edits made on Wikipedia pages [2] obtained from [114]. This dataset contains 1000 items, $10,000$ most active users, resulting in $672,447$ interactions.

- **Reddit post dataset:** We processed reddit [3] forum dataset, which consists of one month of posts made by users. We first samples 1000 most active reddit post and the users who made at least 5 posts on the selected posts. This resulted in $13,840$ users and a total of $121,258$ interactions.

- **StackOverFlow dataset:** We also gathered data from the popular question-answering website, StackOverFlow[4] . For this dataset also, we extracted users who made at least 5 posts. There are $4,125$ users and $20,719$ posts in this dataset.

These datasets, in addition to varying in size of users and density of interactions, also comprise of different users' behavior in terms of repetitive item consumption. In Wikipedia, Reddit, and StackOverFlow a user interacts with the same item consecutively in 79%, 77% and 62% interactions, respectively.
Code available at `https://github.com/shalini1194/IACN`.
Metrics. We evaluate forum recommendation performance using the mean reciprocal rank (MRR) and recall@10. MRR is a standard ranking metric formulated as:

[2] `https://meta.wikimedia.org/wiki/Data_dumps`.
[3] http://files.pushshift.io/reddit/
[4] https://archive.org/details/stackexchange

$MRR = \frac{1}{\text{rank}_{\text{pos}}}$, where rank_{pos} denotes the rank of positive item. Recall@10 is the fraction of ground truth items ranked in the top 10 recommended items.

Comparison Approaches. To verify the performance gain of IACN, we compare its performance with various state-of-the-art models which can be categorized into four classes:

1. RNN based models: This category comprises of RNN based models such as LSTM [122], RRN [106] among others. RNN uses only static embeddings to represent items and predicts users' embedding based on the items they have interacted with. RRN is widely used method and generates dynamic user and item embeddings based on the item and user interaction sequence independently. Both these models take one-hot vector of items as inputs.

2. Co-evolutionary models: These models update both user and item embedding when a user interacts with an item. We compare our model with JODIE [114] and Deep Co-evolve [41]. Both the models use RNN to learn representations of users and items. Deep-Coevolve uses the point process technique to predict the intensity of interaction between user and item, while JODIE uses Euclidean distance between the learned representation to predict the next item to recommend.

3. Temporal Network Embedding: Temporal Network Embedding models are used to generate embedding of nodes of a temporal network. HTNE [117] is a state-of-the-art model for temporal network embedding which integrates the Hawkes process into network embedding so as to capture the influence of historical neighbors on the current neighbors

4. Social Network: We compare our method with GraphRec [123] that combines the information from social network and interaction network to predict user embedding. However, it does not consider the temporal nature of the setting.

5.7.1 Performance Comparison (RQ1)

Table 5.5 compares the performance of IACN with the six state-of-the-art methods. We make the following observations from the results. IACN significantly outperforms all baselines in all datasets across both the metrics. GraphRec performs better than

Table 5.5: Performance comparison on four datasets for all methods. The best and the second best results are highlighted by **boldface** and underlined respectively. Gain% denotes the performance improvement of IACN over the best baseline.

Methods	Wikipedia		Reddit		StackOverFlow	
	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10
LSTM [122]	0.329	0.455	0.205	0.251	0.014	0.017
RRN [106]	0.522	0.617	0.290	0.312	0.019	0.019
HTNE [117]	0.500	0.624	0.211	0.313	0.100	0.178
GrapRec [123]	0.634	0.823	0.621	0.815	0.012	0.041
DeepCo-evolve [41]	0.515	0.563	0.271	0.405	0.017	0.019
JODIE [114]	0.746	0.822	0.755	0.919	0.058	0.063
IACN	**0.796**	**0.861**	**0.869**	**0.922**	**0.106**	**0.280**
Gain %	6.702	4.617	15.099	0.326	6.000	57.303

Table 5.6: Ablation analysis.

Methods	Wikipedia		Reddit		StackOverFlow	
	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10
IACN - Influence	0.776	0.833	0.717	0.919	0.050	0.059
IACN-Attention+RNN	0.786	0.848	0.717	0.92	0.056	0.059
IACN-Fusion+LatentCross	0.612	0.776	0.702	0.918	0.072	0.012
IACN	0.796	0.861	0.869	0.922	0.106	0.280

HTNE for Reddit and Wikipedia dataset. We believe that one of the reasons is the high volume of interactions in less timespan for these datasets. Due to this, the effect of time intervals between interactions is not observed here. HTNE models the impact of time intervals between interactions, which results in its better performance for StackOverFlow compared to GraphRec. We find that for StackOverFlow dataset HTNE performs better than JODIE. This can be attributed to the idea that user-user affinity is more pronounced due to peer-engagement and depth of discussion on these platforms [124]. The fact that IACN outperforms co-evolutionary models confirms our hypothesis that it is important to consider both influence-based and interaction-based signals to predict embedding of user.

5.7.2 Analysis of IACN (RQ2)

Table 5.6 shows the performance comparison of variation of IACN. We describe the variants and discuss the result drop caused by them:

IACN-Influence: Removing the influence modeling layer results in a co-evolutionary model with attention mechanism to update the embedding. We find that removing the influence modeling layer results in drop of IACN performance, revealing that it is useful to model the influence of other users on user interest evolution.

IACN-Attention+RNN: In this variant, we replace the attention in the interaction modeling layer with RNN. The drop in performance indicates that attention mechanism is better able to predict the embedding of user and item by adaptively assigning weights to the past interactions.

IACN-Fusion+LatentCross In this variant of IACN, we replace our Fusion layer with LatentCross [113]. Essentially, we take an element-wise product of user embedding $u(t)$ and the time context vector, $\boldsymbol{w}_t = \boldsymbol{w} * \Delta$, where, $\boldsymbol{w}$ is initialized by 0-mean Gaussian function and Δ is the elapsed time since user's last interaction. Then, we add the influence-based embedding to the resultant vector.

$$\boldsymbol{u}(t+\Delta) = (\mathbf{1} + \boldsymbol{w}_t) * \boldsymbol{u}(t) + \boldsymbol{I}_u(t+\Delta)$$

Using LatentCross instead of our fusion layer degrades performance of IACN showing that fusions layer is better then LatentCross.

The conclusion of this chapter is described in Chapter 7

Chapter 6

Goal Understanding: Learning Trajectory Recommendation

MOOCs provide very diverse set of courses and courses with the same course topic but taught by different teachers. It becomes difficult for students who want to acquire a skill to find the relevant courses. Moreover, a course consists of a number of video lectures, with each one covering some specific knowledge concepts and a student might not be interested in the entire course but certain knowledge concepts taught in the course. To improve student experience, it is important to understand the goal of student for enrolling in the MOOC and then recommend him the relevant next activity. Various challenges are involved in the task of recommending the next activity to students. First, it is important to understand the student goal and then recommend the next activity to fulfill that goal. However, student goal is not known explicitly and can only be inferred from the student behavior. Second, student goal is dynamic and we need to keep track of the student goal at every time instance to recommend him the relevant activities. Third, MOOCs attract diverse group of students with different backgrounds and learning preferences. While recommending the next activities, we need to take into account their preferences and navigation styles.

To understand and capture student interests on MOOCs paltforms, efforts have been put, such as, course recommendation [125, 82], behavior prediction [126] and intention understanding [127] and knowledge concept recommendation [66]. However,

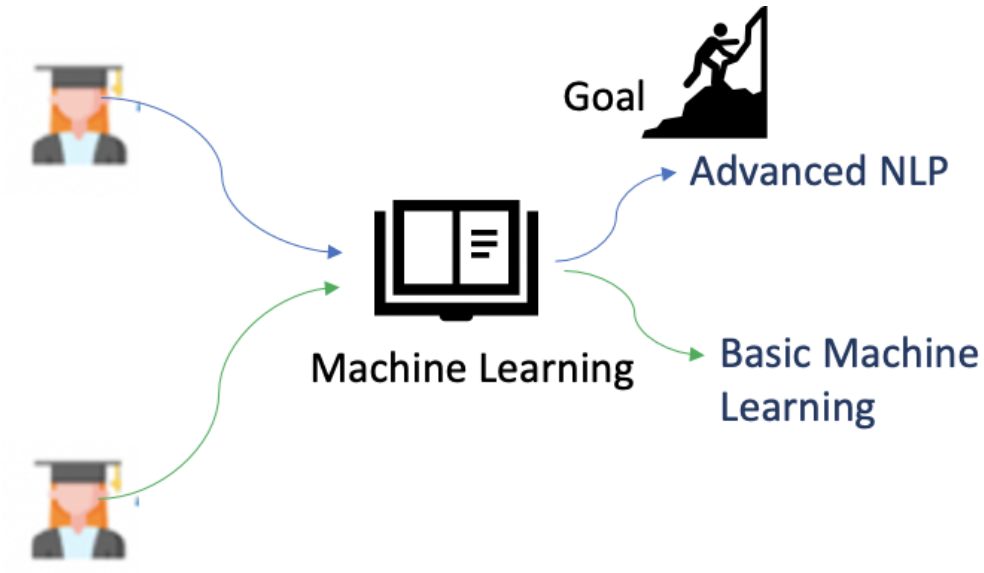


Figure 6.1: Students with different goals but taking the same course.

these models capture student interest at course-level ignoring the fact that students on MOOCs are usually interested in learning certain topics of each course rather than the entire course [1, 66]. In order to address this issue, [66] proposed recommending student the next course topic to study. However, their model fails to capture the dynamic student interest. In addition, while developing a next activity recommender system for students domain factors have to be taken into account.

First, students enrolls in a course with different motivations such as, learning basic ML or learning NLP. For example, as shown in fig 6.1,. The motivation of students can be inferred from their interaction behavior. However, only relying on sequential interaction data of students might be insufficient to capture the motivation of diverse groups of students. Second, in order to motivate students study a new course topic, we need to provide an explanation such as "Study Integral Calculus because you have studied Differential Calculus". This motivates us to build an explainable recommender system for next video recommendation.

6.1 Theoretical Framework

Models of self-regulated learning emphasize that learners are more effective when they take a purposeful role in their own learning, including selection of courses that align with their learning and professional goals [94, 93]. For a system to adequately provide access to such information, it is important to provide self-learners further information about the concept pre-requisite relations and recommend them the next course topic to learn.

6.2 Meaningful Learner Profiling

A core component of most theories of learning is that learners' characteristics influence the ways and the extent to which they learn. Personalized learning, in principle, takes into account learner characteristics to optimize their learning. It follows that for learning to be personalized, information about learners, derived from their behavioral data, must be used to adapt the design of the learning environment and enhance learning outcomes. One fruitful approach in personalizing learning is the identification of different profiles of learners, so that their needs are addressed by course design. These profiles can provide insights into the actual cognitive (e.g., which strategies are used), meta-cognitive (e.g., the conditions under which strategies are used), and motivational processes (e.g., the intensity by which strategies are used) learners engage in during online learning. Understanding what unique interactions between the learner and the course result in these profiles, as well as the stability of these profiles across time and across courses or domains can help further refinement of both learning theories and course design [128, 129, 130].

learners progress through learning on their own pace, guided by their learning goals and interests, and engaging with course content in ways that are consistent with those goals and interests [128]. Additionally, learners vary in the actual processes they engage in during learning [94]. In this work, we propose to understand learner characteristics and learning processes based on their interaction with the learning platform. We will identify distinct interaction sequence patterns and match those sequence patterns to learning goals and processes (cognitive, meta-cognitive, and motivational processes).

Table 6.1: Types of students and their characteristics

ID	Type name	Activity Pattern
0	Strategic but selective	Each video is watched multiple times but a subset of videos are watched and the level of engagement remains high from beginning to end
1	Nonstrategic, non-engaged	The engagement of these learners is fairly low at the onset, and keeps dropping over time with the low number of watches per video
3	Strategic but not engaged learners	Engagement level is decent at the onset but eventually feel challenged or bored leading to lower engagement by the end .
4	Strategic and engaged	Each video is watched multiple times and engagement with each video remains high from beginning to end.

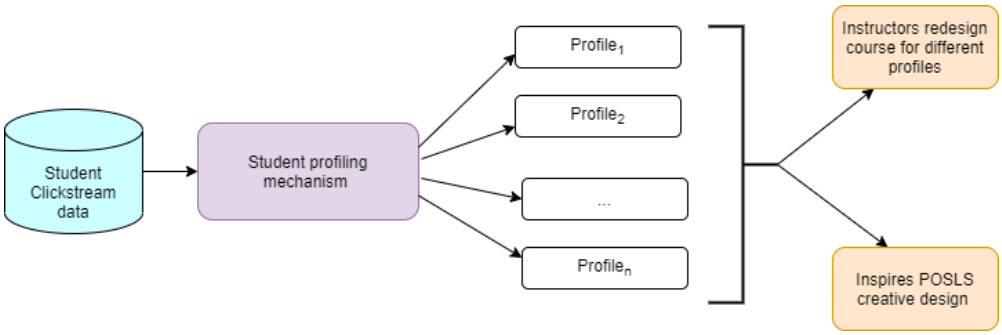


Figure 6.2: Learner Profiling based on their activities

Building on the learner profiling mechanism, we develop a system to profile different learner based on their learning pattern. We characterize these profiles from the learning theory background. Essentially, we first clustered learners based on the features we extracted from their interaction with the learning platform. Then we map those clusters to the different patterns interaction sequences originating from the learning theory perspectives. With this, we come up with four cohorts of learner types, which is also automatically discovered by our algorithm without any prior knowledge. Looking into the learner clusters, we find their different combinations of features quite meaningful. Subsequently, we are able to give the learner types intuitive names, which are shown in Table 6.1. As shown in Figure 6.2, we plan to use these recognised cohorts of learners to improve POSLS by adopting different policies for different cohorts. With these designed cohorts a new learner can be profiled early on in the lecture. With this information, we can improve several tasks such as, drop out prediction, course content recommendation, identify if learner is bored or challenged.

6.3 Representation Learning for POSLS

For developing personalized system, representation learning presents a powerful technique that learns embeddings to represent MOOC entities. These representation can then be used for various downstream tasks.

Learning comprehensive representations for MOOC entities which can directly be employed in various downstream tasks remains a challenge. Most previous work is task-oriented, and directly aim to obtain representation using hand crafted features

[59, 58, 131, 67] obtained from the structure of organization or learn representation from the textual content [132, 61]. However, these models mostly require large amount of training labels (e.g., concept pre-requisite labels). To tackle this challenge, in this paper, we investigate the problem of learning an overall representation of MOOC entities in an unsupervised manner. Our goal is to demonstrate that these pre-trained embeddings can improve various downstream tasks.

Several pre-training methods have shown their superiority in Natural Language Processing (NLP) on representation learning task [70, 69]. However, they only exploit the textual content of the entities. Simply employing these methods to learn MOOC entity representation is not sufficient as we show in our work. Our method MERIT [1] incorporated the inherent relations between MOOC entities. These relations provide richer information about them and hence enhance their representation. For example, as shown in Figure 6.3 a course consists of sequence of topics, each topic consists of sequence of videos. In addition, each video is annotated with the corresponding universal concepts. All this information are crucial for understanding the MOOC entity, which requires us to find an appropriate way to aggregate them for learning a comprehensive representation. Second, the organization of a course with respect to concepts taught in each video presents the domain knowledge about difficulty of concepts. We further utilize this difficulty information to enhance the representation effectiveness.

To evaluate our pre-trained entity embeddings, we show that the learned representations substantially outperform the state-of-the-art models on the important EDM tasks. The first task is predicting the pre-requisite relations between the MOOC concepts. The second task is to tag the concepts to some external source material so that they can be provided to students. Experimental results on the two downstream tasks show the applicability and strength of our model. To summarize, our main contributions in this paper are as follows:

- We are the first to propose a model for MOOC entity representation learning that captures, textual content of these entities, their structural relations, and domain knowledge about concept difficulty. The learned pre-trained embedding can be directly employed to various downstream tasks.

[1] MERIT : **M**OOC **E**ntity **R**epresentation using Graph- **I**nformed **T**ransformers

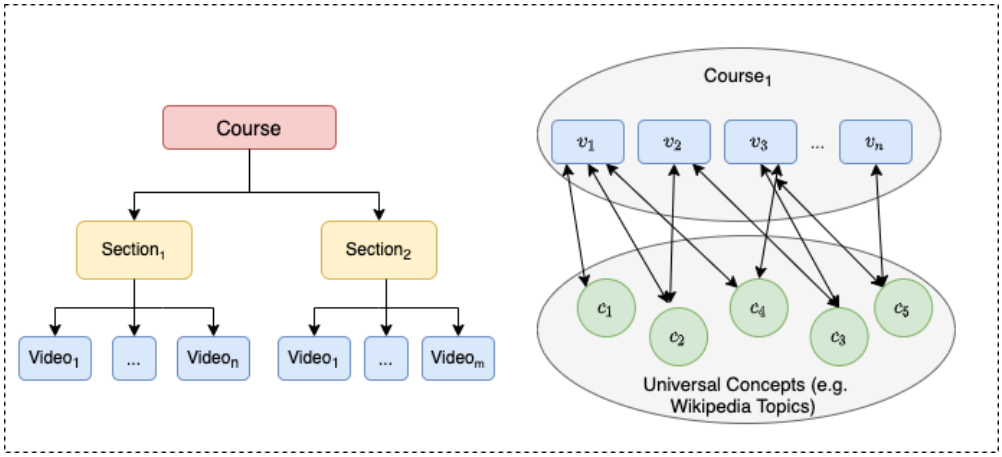


Figure 6.3: Overview of MERIT model: Leftmost figure shows entity's textual content and hierarchical structure in a course and the relation between concepts and videos which constitute the input of MERIT.

- We publicly release the pre-trained embeddings of universal concepts and MOOC videos. These can be used to improve various downstream tasks. We showed the improvement of pre-trained embeddings on concept pre-requisite prediction and video recommendation tasks.
- We perform extensive experiment and show that our pre-trained embedding achieves on an average 7.29% improvement over state-of-the-art pre-training algorithms.

6.3.1 MERIT: A Unified Representation of MOOC Entities using Graph-Informed Transformer

Problem Formulation

In this subsection, we formally introduce the problem and clarify mathematical symbols in this paper. Firstly, a course has a natural hierarchical structure. A course usually consists of multiple topics, $\{T_1, T_2, \ldots, T_n\}$, and each topic consists of sequence of videos, $\{V_1, V_2, \ldots V_m\}$. For example, a course on "Computer Vision Basics" has three topic: "Color, Light, and Image Formation", "Low-, Mid- and High-Level Vision ", "Mathematics for Computer Vision". The "Color, Light, and Image Formation topic"

has sequence of videos: "Light Sources", "Pinhole Camera Model" and "Color Theory". Further, each course video is cross linked with various concepts $\{C_1, C_2, \ldots, C_l\}$ where each video covers multiple concepts and a concept can be covered by multiple videos. For example, a video on "Mathematics for Computer Vision" is linked with the concept "Basic Convex Optimization" and "Basic Convex Optimization" can be associated with other videos such as "Theory of Machine Learning". In addition to these relation information, the textual content consisting of name and description of each entity is also available.

Our goal is to learn task-independent and effective representations of all the above described MOOC entities, i.e., the output, encode the individual entity feature along with different relations between them as a single vectors. In the following sections, we will address the main three challenges: (1) how to generate individual entity representation; (2) how the representation is pre-trained; (3) how the representation is applied to downstream tasks.

Model Architecture

We will now describe each layer of MERIT.

Pre-trained BERT to encode the textual content

Firstly, we encode concepts using its textual content which consists of its name and description. We employ a pre-trained BERT-base model to obtain the representation of concepts. Similar to the procedure employed for document representation [133] the final representation of the [CLS] token is used as the output representation of the entity.

$$\boldsymbol{x} = \text{BERT}(input)_{[CLS]}, \tag{6.1}$$

where BERT is the pretrained- BERT model [2] , and input is the concatenation of the [CLS] token and WordPieces [134] of the name and description of the entity. We use the pre-trained BERT as our model initialization because it has been trained on large Chinese simplified and traditional text. Using this model, we learn the initial representation of concepts ($\boldsymbol{c}^0$) and videos ($\boldsymbol{v}^0$).

[2] `https://huggingface.co/bert-base-chinese`

Encoding course structure with Segment-aware Transformers

Input Embedding Layer. As shown in figure 5.2, a course consists of an ordered sequence of topics which consists of videos. We need to encode this hierarchical structure information present in a course in the learned representations. To obtain an embedding of jth video in a course c, we first obtain the corresponding video representation using Equation (1). Additionally, we define a *video position* embedding matrix as $\mathbf{E}^{vp} \in \mathbb{R}^{l \times d}$ to encode the order of videos belonging to a course, where l is the maximum number of videos in a course and *topic position* embedding matrix as $\mathbf{E}^{tp} \in \mathbb{R}^{t \times d}$ to encode the position of topic in the course, where t is the maximum number of topics in the course. This idea is motivated by segment-aware BERT [135] which learns paragraph index, sentence index, and token index embeddings to encode a document's hierarchical structure. Employing video position and topic position encoding is important in learning video and course representation because it inherently encodes the information of organization of the course. This implicit information helps identify the relation between videos of the same course and the contribution of each video in learning the final course embedding.

Afterward, we feed the inputs to transformer layer, and these inputs should convey the representation of videos, their positions in the course and the position of their topic. Thus, the video embedding is obtained as:

$$\hat{\boldsymbol{v}}_i = \boldsymbol{v}_i + \boldsymbol{E}_i^{VP} + \boldsymbol{E}_i^{TP}, \tag{6.2}$$

where $\boldsymbol{v}_i$ is obtained from Equation(1).

Finally, the input video sequence is expressed as $\hat{\mathbf{V}} = [\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \dots \hat{\mathbf{v}}_n]$ by combining the video embedding, the video position embedding and the topic position embedding.

Transformer Layer To learn a course embedding, we take the input video sequence $\hat{\mathbf{V}}$ and add a special token- [CLS] at the end to represent the whole course. Then we use transformer layer [38] to encode the entire sequence. The transformer layer is composed of two sub-layers:

$$\boldsymbol{h}_l = \text{LayerNorm}(\boldsymbol{z}_{l-1} + \text{MHAtt}(\boldsymbol{z}_{l-1})), \tag{6.3}$$

$$\boldsymbol{z}_l = \text{LayerNorm}(\boldsymbol{h}_l + \text{FFN}(\boldsymbol{h}_l)), \tag{6.4}$$

where LayerNorm is a layer normalization proposed in [98] ; MHAtt is the multihead attention mechanism introduced in [38] which allows each token to attend to other tokens with different attention distributions; and FFN is a two-layer feed-forward network with ReLU as the activation function. We take the course [CLS] token representation output by the last layer of the global transformers to represent the course semantic and structural features, denoted as z_c.

Pre-training

Graph based pre-training objective

The graph built between videos, concepts and courses, shown in Figure 5.2, can help in identifying the related entities. To encode this relatedness signal, we design objective functions to train the MERIT model so that related entities lie closer in the embedding space. Particurarly, our MOOCCube graph from [5] consists of vertices of types concepts ($\mathcal{C}$), videos ($\mathcal{V}$) and courses ($\mathcal{Z}$). The edges connecting vertices in this graph can be of two types:

- **Explicit relation** In the video-concept and course-concept bipartite graphs, edges exist between videos and concepts and course and concepts, presenting an explicit signal as it is directly available in the dataset.
- **Implicit relation** This relation indicates the similarity between the entities of the same type and can be induced from provided graph but are not provided explicitly. Specifically, if the number of adjacent nodes between two vertices increases by a threshold, then we consider an implicit relation between those vertices.

For learning representation of each node in the graph, MERIT optimizes a margin-based ranking objective between each edge e in the training data and a set of edges e' constructed by corrupting e.

$$\mathcal{L}_{triplet} = \sum_{e \in \mathcal{G}} \sum_{e' \in S(e)'} \max(f(e) - f(e') + \lambda, 0) \tag{6.5}$$

where λ is a margin hyperparameter, f is the cosine similarity between the two embeddings and

$$S(e') = (s, d') | d' \in type(d), \tag{6.6}$$

where $type(d)$ returns the type of vertex d (concept, video, or course).

Domain-Oriented Objective

The above objectives and features have helped learn the information present in the textual content and the structural relations between entities. However, it is also important to consider the domain knowledge to learn effective representations of these entities. The difficulty of understanding of a concept or a lecture contains important information regarding the concept or lecture. In order to also include such information in final representation, in this section, we designed a domain-oriented objective for our pre-training method. Specifically, the complexity of different concepts captures the domain-specific knowledge. Different concepts have different complexities and this complexity level is inherent in their distributions in the course. Specifically, for a concept in MOOCs, if it covers more videos in a course or it survives longer time in a course, then it is more likely to be a basic concept rather than an advanced one [59]. We then use the following formal equations to compute average video coverage (avc) and the average survival time (ast) of a concept i as follows,

$$avc(i) = \frac{1}{|\mathcal{C}(i)|} \sum_{\mathcal{C} \in \mathcal{C}(i)} \frac{|I(C,i)|}{|C|}, \tag{6.7}$$

$$ast(i) = \sum_{\mathcal{C} \in \mathcal{C}(i)} \frac{|max(I(C,i)) - min(I(C,i)) + 1|}{|C|} \tag{6.8}$$

These two metrics capture the difficulty of a concept. To preserve the difficulty information effectively, for the concept, we use a linear layer to map the activation $\boldsymbol{e}_i$ to a difficulty approximation $\hat{d}_i = \boldsymbol{w}^T \boldsymbol{e}_d i + \boldsymbol{b}_d$ where $\boldsymbol{w}_d$ and $\boldsymbol{b}_d$ are network parameters. We use the concept difficulty $\boldsymbol{d}_i$ as the auxiliary target, and design the following loss function $\mathcal{L}$ to measure the difficulty approximation error:

$$\mathcal{L}_{mse} = \sum_{i=1}^{C} ||d_i - \hat{d}_i||_2. \tag{6.9}$$

To generate entity embeddings that preserve explicit relations, implicit similarities, and concept difficulty simultaneously, we combine all the loss functions together and minimize the following loss:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{triplet} + (1 - \lambda_1)\mathcal{L}_{mse}, \tag{6.10}$$

where, λ_1 is a tunable hyperparameter. After pre-training, MERIT entity representation should be able to capture both relation and individual features, and transfer the understanding of these entities to downstream tasks in the area of education.

6.3.2 MOOC Entity Representation Evaluation

After learning the representation of MOOC entitiess, we employ them to improve downstream tasks in the area of education. The first task we focus on is concept pre-requisite prediction task. for example, as in the research by Pan et al. [59], the authors use hand-crafted features followed by classification methods. Another task important for education community is video recommendation [66]. In the paper of [60], each video is represented as a single vector and then serves as the input to sequence models to predict the next element of sequence.

To apply MERIT representation to a specific task, we just provide the required representation to replace the equivalent part of the downstream model, which minimizes the cost of model modification. By doing this, we provide a better initialization to the downstream model, leading to their faster convergence and better optimization.

In summary, MERIT has the following advantages for MOOC entity representation learning. First, it provides a unified and universally applicable representation for MOOC entities. Second, it is able to incorporate both textual content and structural relation between entities along with the domain knowledge about concept difficulty. Third, it is easy to directly apply them on various downstream tasks.

6.4 Experimental Settings

In this section, we present our experimental settings to answer the following questions:
RQ1 Can MERIT outperform the state-of-the-art methods for Concept Pre-requisite Prediction task?
RQ2:Can MERIT outperform the state-of-the-art methods for Lecture Recommendation task?
RQ3: What is the influence of various components in the MERIT architecture?

Table 6.2: Dataset Details

	Mathematics	Computer Science
#Concepts	373	496
#Courses	459	190
#Lectures	10,308	5,085
# Lectures/Course	22.46	26.76
#Tokens/Concept	34.17	54.24
#Tokens/Lecture	1365.73	1378.94
#Concepts/Course	17.85	22.84
#Concepts/Lecture	1.93	18.19
# Concept Pre-requisites	1,314	1,604
#Users	37,849	26,588
#Lecture/User	60	67.79
%Repetitive elements in interactions	0.015	0.011
%Sequential elements in interactions	62.1	66.7

6.4.1 Dataset

The dataset we used for our experiments have been obtained from the online education system called XuetangX and publicly available in [5]. This dataset consists of concepts along with their description from Wikidata, video playlists from a MOOC corpus along with the subtitles of the videos, courses where each course consists of several topics and each topic is covered by several videos. These videos are annotated with set of associated concepts . We consider the data from two domains, Mathematics and Computer Science. The details for the dataset are shown in Table 6.2.

There are total 373 and 407 concepts, 459 and 190 courses , and $10,308$ and $5,085$ lectures in Mathematics and Computer Science, respectively. On average, in Mathematics domain each course has 22.46 lectures and 17.85 concepts and each lecture has 1365.73 tokens; while each concept has 34.17 tokens. In Computer Science domain each course 26.76 lectures and 22.84 concepts; while each lecture has 1378.94 token and each concept has 54.24 tokens.

Since the MOOC concepts are universal and our goals is to learn pre-trained embeddings of these concepts, we augment the concepts from other available datasets [59, 63, 58] to our dataset. This results in total $1,314$ pre-requisite relations for Mathematics concepts, while $1,604$ for Computer Science concepts. For lecture recommendation, we considered the user interaction with courses of Mathematics and Computer Science domain only. This results in $37,849$ and $26,588$ users, and 60 and 67.79 interactions per user on average for Mathematics domain and Computer Science domain, respectively. In order to verify that learners do not necessarily follow the sequence of lectures set by instructors, we also compute the percentage of times consecutive interaction patterns occur in the set sequence by instructors. We find that it only occurs 62.1% and 66.7% times in Mathematics and Computer Science dataset, respectively.

6.4.2 Evaluation Tasks

We employ state-of-the-art supervised learning methods for concept pre-requisite prediction [62] and lecture recommendation [60]. To evaluate the effectiveness of MERIT, we compare the quality of our pre-trained embeddings with pre-trained embeddings learned

from the competing approaches. All these methods are able to generate entity representation, and then be applied to the two models mentioned above as warm-initialization. Specifically, these methods are:

- **Random:** We randomly assign the embeddings which essentially results in the original supervised models.
- **Word2vec:** Assign text tokens with the corresponding word2vec embedding [136] and CLS token as input to LSTM layer and use the embedding of [CLS] token as the entity embedding.
- **Doc2vec** Similar to word2vec but with an additional paragraph vector to learn document representation [73]
- **BERT** is a state-of-the-art pre-training method featuring bi-directional transformer layers and masked language model [70].
- **PBG** is an embedding system that takes only the MOOC graph as input and learns entity representation in an unsupervised manner [75]. We specifically, employ TransE [137] model to learn the representations.

6.4.3 Implementation Details

For each evaluation tasks, we split the dataset into 80%, 10%, and 10% as training, validation, and test set. We take pretrained BERTbase with 12 layers to encode local semantic features from lectures and concepts. Pretrained model weights are obtained from Pytorch transformer repository[3] . Besides, we set the number of global transformer layers as 2 based on the preliminary experiments. We find the 2 layer global transformer layers work much better than 1 layer. All transformer-based models/layers have 768 hidden units. The model is trained on Nvidia GeForce GTX 1050 Ti GPU. The optimizer is Adam [138] with learning rate of $1e-5$. We set the epochs as 200, batch size as 64. For word2vec we used pre-trained word vectors from [139] and words are extracted using jieba [4] . For doc2vec as well, we used the pre-trained word vectors from above

[3] https://huggingface.co/bert-base-chinese
[4] `https://github.com/jsrpy/Chinese-NLP-Jieba`

Table 6.3: Performance comparison on concept pre-requisite prediction task. The best performing method is boldfaced, and the second best method in each row is underlined. Gains are shown in the last row.

	Mathematics			Computer Science			
	P	R	F1	P	R	F1	Avg
Random	0.583	0.594	0.587	0.528	0.571	0.545	0.568
Word2vec	0.630	0.638	0.634	0.570	0.577	0.573	0.604
Doc2vec	0.645	0.644	0.642	0.594	0.606	0.599	0.622
BERT	<u>0.646</u>	<u>0.660</u>	<u>0.652</u>	<u>0.630</u>	<u>0.616</u>	<u>0.621</u>	<u>0.632</u>
PBG	0.636	0.626	0.630	0.607	0.611	0.609	0.620
MERIT	**0.701**	**0.685**	**0.692**	**0.659**	**0.670**	**0.663**	**0.678**
Gain %	8.557	5.771	6.162	8.567	9.656	8.867	7.930

in addtion to gensim APIs to obtain document embedding [5] . For modeling transE, we used Pytorch BigGraph [75] to obtain entity embeddings with embedding size of 768.

6.5 Results and Discussion

Our evaluation of MERIT's pretrained entity representations on the two downstream tasks is shown in Table 6.3. Overall, we observe substantial improvements across both the tasks with average performance of 0.656 across all metrics on all tasks which is a 8.34% relative improvement over the next-best baseline. We now discuss the results in detail.

6.5.1 Concept Pre-requisite Prediction (RQ1)

For concept pre-requisite prediction, we used PREREQ [62] as the base model and initialized the embeddings with pre-trained embeddings from baseline methods and our MERIT model. Since concept pre-requisite prediction is a binary classification task, where given a pair of concepts (a, b), the task is to predict whether a is pre-requisite

[5] https://www.tutorialspoint.com/gensim/gensim_doc2vec_.htm

Table 6.4: Performance comparison on lecture recommendation task. The best performing method is boldfaced, and the second best method in each row is underlined. Gains are shown in the last row.

	Mathematics			Computer Science			
	HR@10	NDCG	MRR	HR@10	NDCG	MRR	Avg
Random	0.417	0.297	0.278	0.372	0.262	0.247	0.312
Word2vec	0.458	0.358	0.344	0.438	0.325	0.298	0.370
Doc2vec	0.598	0.476	0.451	0.556	0.427	0.400	0.485
BERT	0.650	<u>0.598</u>	0.458	0.628	0.534	0.485	0.559
PBG	<u>0.654</u>	0.592	<u>0.552</u>	<u>0.643</u>	<u>0.559</u>	<u>0.502</u>	<u>0.584</u>
MERIT	**0.683**	**0.615**	**0.604**	**0.679**	**0.614**	**0.605**	**0.633**
Gain %	4.404	2.809	9.402	5.552	9.857	20.438	8.744

Table 6.5: Ablation Study on Concept Pre-requisite Predition task.

	Mathematics			Computer Science		
	P	R	F1	P	R	F1
Without concept difficulty	0.657	0.671	0.662	0.649	0.632	0.639
Without explicit similarity	0.664	0.666	0.665	0.647	0.634	0.640
Without implicit similarity	0.656	0.682	0.665	0.636	0.626	0.630
Without explicit+implicit	0.649	0.664	0.653	0.623	0.647	0.638
MERIT	**0.701**	**0.685**	**0.692**	**0.659**	**0.670**	**0.663**

Table 6.6: Ablation Study on lecture recommendation task.

	Mathematics			Computer Science		
	HR	NDCG	MRR	HR	NDCG	MRR
Without concept difficulty	0.670	0.607	0.598	0.665	0.608	0.601
Without explicit similarity	0.677	0.608	0.597	0.660	0.602	0.594
Without implicit similarity	0.664	0.599	0.589	0.651	0.594	0.586
Without explicit+implicit	0.656	0.589	0.579	0.642	0.593	0.580
MERIT	**0.683**	**0.615**	**0.604**	**0.679**	**0.614**	**0.605**

of b, we report Precision (P), Recall (R) and Macro-averaged F1 score (F1) to compare the different models. We observe that PREREQ performance when trained on our representations is better than when trained on any other baseline. Particularly, on the Computrer Science (Mathematics) dataset, we obtain an 0.692(0.663) F1 score which is about a relatively 6.16%(8.87%) relative improvement over the best baseline on each dataset respectively.

6.5.2 Lecture Recommendation (RQ2)

For lecture recommendation, we used the method proposed in [60] as base model which employs an LSTM to predict user's next lecture. We initialize the lecture embeddings with those pre-trained from the baseline models and MERIT model. We use ranking metrics to evalaute the recommendation performance, specifically report HR@10, nDCG@10 and MRR. We observe that MERIT outperforms all the baseline models on this task as well. For Computer Science dataset (Mathematics), MERIT achieves $0.679(0.693), 0.611(0.617)$, and $0.600(0.603)$ at HR@10, nDCG@10, and MRR, respectively; MERIT outperforms by $3.84\%(7.78\%), 2.19\%(10.38\%)$, and $8.77\%(20.12\%)$ relative improvement over second best baseline on each dataset respectively.

Another observation is that BERT model which captures only the semantic relation between entities is the second best model for concept pre-requisite prediction; while PBG which captures the structural relation between entities is the second best model for lecture recommendation. This can be attributed to the fact that the textual content

of concepts is obtained from Wikipedia [5] and hence the learned embeddings are good quality. On the other hand, lectures' textual content is noisy resulting in BERT not able to generate their good quality embeddings. Our model, utilizing both textual content and structural relation between entities by taking advantage of the best of both worlds performs superior than both PBG and BERT.

6.5.3 Ablation Study (RQ3)

To get deep insights on the MERIT model, we investigate the contribution of various components involved in the model. Therefore, we conduct some ablation experiments to show how each part of our method affect final results. In Table 6.5 and 6.6, there are following variations of MERIT, each of which takes out one component from the full model.

- **Without explicit similarity** In this variant, weremove the explicit similarity objective when training the network. Specifically, we remove the course-concept and lecture-concept similarity loss.
- **Without implicit similarity** In this variant, we remove the loss function resulting from the implicit similarity between entities. Specifically, we remove the concept-concept, lecture-lecture, and course-course similarity loss.
- **Without explicit+implicit** In this variant, both the explicit and implicit similarity loss. The network is only trained with the BERT model followed by concept difficulty loss.
- **Without difficulty constraint** In this variant, we remove the loss function resulting from the difficulty prediction of a concept. The network is only trained with BERT model to encode the textual content and the the structural information but not the domain knowledge involved in predicting the concept difficulty.

The result in the above tables indeed shows many interesting conclusions. Removing individual component does reduce the performance of methods on both concept prerequisite prediction and lecture recommendation tasks.

First the removal of loss resulting from explicit and implicit relations causes the most decline in the performance. Thus, incorporating the structural relations between

entities is the most important factor for MERIT. Second, the models show a similar degree of decline when removing explicit and implicit similarities, which means these two pieces of information are equally important. The domain knowledge also plays a significant contribution to the model performance for concept pre-requisite prediction task; while not that much for the lecture recommendation task. The importance of concept difficulty for concept pre-requisite prediction task is that, for one concept to be pre-requisite of the other, it is important that they are both related and one is more difficult than the other. For lecture recommendation, the MERIT model already takes into account the position of lecture in the course which already encodes the difficulty relations between lectures in the same course. This information along with learner interaction information provided as training example gives sufficient information for the lecture recommendation model to learn the difficulty relations between lectures and their associated concepts. Thus explicit concept difficulty does not inform the lecture recommendation task significantly.

The concluding remarks are present chapter 7

Chapter 7

Conclusion

In this proposal, we described our design Personalized Online Self-learning System (POSLS). We took steps towards building POSLS, specifically in the field of knowledge assessment, interest prediction and goal understanding.

We describe the various applications and modeling layer required for a fully developed POSLS in chapter 3. We also discuss how POSLS can help alleviate the existing issues with MOOC. In addition, we describe the various benefits the developed platform will have for both learners and teachers. The design goals are an augmentation to the existing MOOC platform to maintain learner's engagement and

In Chapter 4, we proposed a self-attention based knowledge tracing model. It models a student's interaction history (without using any RNN) and predicts his performance on the next exercise by considering the relevant exercises from his past interactions. In the future, we plan to further investigate the learning and forgetting curve of a student while going through a sequence of learning activities and incorporate those techniques in our model. We delivered a Self-attention based models for KT. It models a student's interaction history and predicts her performance on the next exercise by considering contextual information obtained from its relation with the past exercises and the forget behavior of the student. The relation between exercises is computed using the student performance data and the textual content of exercises. The forget behavior is modeled using a time decaying kernel function. The contextual information is then incorporated in a self-attention layer which we call relation-aware self-attention. Owing to the purely self-attention mechanism self-attention based models are interpretable.

Extensive experimentation on a variety of real-world datasets shows that our model can outperform the state-of-the-art methods and is an order of magnitude faster than the RNN-based approaches.

In Chapter 5, we described a student interest trajectory based solution to MOOC thread recommendation problem. Our method, SITRec models the dynamic nature of student interest and thread contents. It also leverages the course topic structure and how student interest towards a thread changes when posts are made on a thread that the student has already interacted with. This captures the temporal dynamics of posting behavior of students in online forums. We demonstrate the superiority of the performance of our model compared to other competing approaches on three real-world datasets. Finally, in chapter 6, we develop models to aid self-learners in taking the most optimal trajectory for gaining a skill. For this, we provide both the concept pre-requisite lists so that learners can decide themselves what concepts they need to focus to gain mastery over their goal concepts. In addition, we also provide actual lecture recommendation for students to watch next based on their history of interactions. We leverages the relations between all the entities of MOOC, their textual content and domain knowledge to learn powerful embeddings of all MOOC entities which can be used for various downstream tasks. The pre-trained embeddings help the downstream models in two folds: augmenting richer information about entities, and generalizing the model for unseen behavior of students. We also utilize the interactions data of student with MOOC lectures to understand the temporal pattern of students with different behaviors and motivations.

To summarize, MOOCs is still in its beginner's mode and there exists enormous scope of research in this field. This proposal transforms MOOCs into test-beds for advancing educational research, and ultimately, improving learning. Personalzied MOOCs framework produces a huge impact on the society by providing a cost-effective way for everyone to afford high quality education. I address the various issues associated with developing personalized online self-learning systems by developing new machine learning models.

Bibliography

[1] Han Yu, Chunyan Miao, Cyril Leung, and Timothy John White. Towards ai-powered personalization in mooc learning. *npj Science of Learning*, 2(1):1–5, 2017.

[2] Nabeel Gillani, Taha Yasseri, Rebecca Eynon, and Isis Hjorth. Structural limitations of learning in a crowd: communication vulnerability and information diffusion in moocs. *Scientific reports*, 4:6447, 2014.

[3] V. Sabnis, P. D. Tejaswini, and G. S. Sharvani. Course recommendations in moocs : Techniques and evaluation. In *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*, pages 59–66, 2018.

[4] Apostolos Koutropoulos and Panagiotis Zaharias. Down the rabbit hole: An initial typology of issues around the development of moocs. *Current Issues in Emerging eLearning*, 2(1):4, 2015.

[5] Jifan Yu, Gan Luo, Tong Xiao, Qingyang Zhong, Yuquan Wang, Junyi Luo, Chenyu Wang, Lei Hou, Juanzi Li, Zhiyuan Liu, et al. Mooccube: A large-scale data repository for nlp applications in moocs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3135–3142, 2020.

[6] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Advances in neural information processing systems*, pages 505–513, 2015.

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

[8] Shalini Pandey and Jaideep Srivastava. Rkt: Relation-aware self-attention for knowledge tracing. *arXiv preprint arXiv:2008.12736*, 2020.

[9] Shalini Pandey, George Karypis, and Jaideep Srivastava. An empirical comparison of deep learning models for knowledge tracing on large-scale dataset. *arXiv preprint arXiv:2101.06373*, 2021.

[10] Shalini Pandey, Andrew Lan, George Karypis, and Jaideep Srivastava. Learning student interest trajectory for moocthread recommendation. *arXiv preprint arXiv:2101.05625*.

[11] Shalini Pandey, George Karypis, and Jaideep Srivasatava. Iacn: Influence-aware and attention-based co-evolutionary network for recommendation. *arXiv e-prints*, pages arXiv–2103, 2021.

[12] Ling Zhang, James D Basham, and Sohyun Yang. Understanding the implementation of personalized learning: A research synthesis. *Educational Research Review*, page 100339, 2020.

[13] William Swartout, Benjamin D Nye, Arno Hartholt, Adam Reilly, Arthur C Graesser, Kurt VanLehn, Jon Wetzel, Matt Liewer, Fabrizio Morbini, Brent Morgan, et al. Designing a personal assistant for life-long learning (pal3). In *29th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016*, pages 491–496. AAAI Press, 2016.

[14] Arthur C Graesser, Xiangen Hu, Benjamin D Nye, Kurt VanLehn, Rohit Kumar, Cristina Heffernan, Neil Heffernan, Beverly Woolf, Andrew M Olney, Vasile Rus, et al. Electronixtutor: an intelligent tutoring system with multiple learning resources for electronics. *International journal of STEM education*, 5(1):15, 2018.

[15] Ivon Arroyo, Beverly Park Woolf, Winslow Burelson, Kasia Muldner, Dovan Rai, and Minghui Tai. A multimedia adaptive tutoring system for mathematics that addresses cognition, metacognition and affect. *International Journal of Artificial Intelligence in Education*, 24(4):387–426, 2014.

[16] Chih-Ming Chen. Intelligent web-based learning system with personalized learning path guidance. *Computers & Education*, 51(2):787–814, 2008.

[17] Steven P Reise. Item response theory. *The Encyclopedia of Clinical Psychology*, pages 1–10, 2014.

[18] Philip I Pavlik Jr, Hao Cen, and Kenneth R Koedinger. Performance factors analysis–a new alternative to knowledge tracing. *Online Submission*, 2009.

[19] Hao Cen, Kenneth Koedinger, and Brian Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International Conference on Intelligent Tutoring Systems*, pages 164–175. Springer, 2006.

[20] Jimmy De La Torre. The generalized dina model framework. *Psychometrika*, 76(2):179–199, 2011.

[21] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[22] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*, pages 171–180. Springer, 2013.

[23] Ryan SJD Baker and Kalina Yacef. The state of educational data mining in 2009: A review and future visions. *JEDM— Journal of Educational Data Mining*, 1(1):3–17, 2009.

[24] Zachary A Pardos and Neil T Heffernan. Kt-idem: Introducing item difficulty to the knowledge tracing model. In *International conference on user modeling, adaptation, and personalization*, pages 243–254. Springer, 2011.

[25] Nguyen Thai-Nghe, Lucas Drumond, Artus Krohn-Grimberghe, and Lars Schmidt-Thieme. Recommender system for predicting student performance. *Procedia Computer Science*, 1(2):2811–2819, 2010.

[26] Andreas Töscher and Michael Jahrer. Collaborative filtering applied to educational data mining. 2010.

[27] Nguyen Thai-Nghe, Tomás Horváth, and Lars Schmidt-Thieme. Factorization models for forecasting student performance.

[28] Chun-Kit Yeung and Dit-Yan Yeung. Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, pages 1–10, 2018.

[29] Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*, pages 765–774, 2017.

[30] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[31] Tiffany Barnes. The q-matrix method: Mining student response data for knowledge.

[32] Qi Liu, Zai Huang, Zhenya Huang, Chuanren Liu, Enhong Chen, Yu Su, and Guoping Hu. Finding similar exercises in online education systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1821–1830, 2018.

[33] Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Hui Xiong, Yu Su, and Guoping Hu. Ekt: Exercise-aware knowledge tracing for student performance prediction. *arXiv preprint arXiv:1906.05658*, 2019.

[34] 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 *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[35] Koki Nagatani, Qian Zhang, Masahiro Sato, Yan-Ying Chen, Francine Chen, and Tomoko Ohkuma. Augmenting knowledge tracing by considering forgetting behavior. In *The World Wide Web Conference*, pages 3101–3107, 2019.

[36] Yuying Chen, Qi Liu, Zhenya Huang, Le Wu, Enhong Chen, Runze Wu, Yu Su, and Guoping Hu. Tracking knowledge proficiency of students with educational priors. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 989–998, 2017.

[37] Hermann Ebbinghaus. Memory: A contribution to experimental psychology. *Annals of neurosciences*, 20(4):155, 2013.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[39] Mingi Ji, Weonyoung Joo, Kyungwoo Song, Yoon-Yeong Kim, and Il-Chul Moon. Sequential recommendation with relation-aware kernelized self-attention. *arXiv preprint arXiv:1911.06478*, 2019.

[40] Baosong Yang, Jian Li, Derek F Wong, Lidia S Chao, Xing Wang, and Zhaopeng Tu. Context-aware self-attention networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 387–394, 2019.

[41] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675*, 2016.

[42] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Learning dynamic embeddings from temporal interactions. *arXiv preprint arXiv:1812.02289*, 2018.

[43] Arti Ramesh, Dan Goldwasser, Bert Huang, Hal Daume, and Lise Getoor. Understanding mooc discussion forums using seeded lda. In *Proceedings of the ninth workshop on innovative use of NLP for building educational applications*, pages 28–33, 2014.

[44] Thushari Atapattu and Katrina Falkner. A framework for topic generation and labeling from mooc discussions. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, pages 201–204. ACM, 2016.

[45] Yumeng Qiu, Yingmei Qi, Hanyuan Lu, Zachary Pardos, and Neil Heffernan. Does time matter? modeling the effect of time with bayesian knowledge tracing. pages 139–148, 01 2011.

[46] Diyi Yang, Mario Piergallini, Iris Howley, and Carolyn Rose. Forum thread recommendation for massive open online courses. In *Educational Data Mining 2014*. Citeseer, 2014.

[47] Fabian Abel, Ig Ibert Bittencourt, Evandro Costa, Nicola Henze, Daniel Krause, and Julita Vassileva. Recommendations in online discussion forums for e-learning systems. *IEEE transactions on learning technologies*, 3(2):165–176, 2009.

[48] Andrew S Lan, Jonathan C Spencer, Ziqi Chen, Christopher G Brinton, and Mung Chiang. Personalized thread recommendation for mooc discussion forums. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 725–740. Springer, 2018.

[49] Christopher G Brinton, Mung Chiang, Shaili Jain, Henry Lam, Zhenming Liu, and Felix Ming Fai Wong. Learning about social learning in moocs: From statistical analysis to generative model. *IEEE transactions on Learning Technologies*, 7(4):346–359, 2014.

[50] Fei Mi and Boi Faltings. Adaptive sequential recommendation for discussion forums on moocs using context trees.

[51] Saman Rizvi, Bart Rienties, Jekaterina Rogaten, and René F Kizilcec. Investigating variation in learning processes in a futurelearn mooc. *Journal of computing in higher education*, 32(1):162–181, 2020.

[52] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Engaging with massive online courses. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, page 687–698, New York, NY, USA, 2014. Association for Computing Machinery.

[53] René F. Kizilcec and Emily Schneider. Motivation as a lens to understand online learners: Toward data-driven design with the olei scale. 22(2), 2015.

[54] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Engaging with massive online courses. In *Proceedings of the 23rd international conference on World wide web*, pages 687–698, 2014.

[55] Andrew S Lan, Christoph Studer, and Richard G Baraniuk. Time-varying learning and content analytics via sparse factor analysis. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 452–461, 2014.

[56] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Engaging with massive online courses. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, page 687–698, New York, NY, USA, 2014. Association for Computing Machinery.

[57] Sara Assami, Najima Daoudi, and Rachida Ajhoun. Personalization criteria for enhancing learner engagement in mooc platforms. In *2018 IEEE Global Engineering Education Conference (EDUCON)*, pages 1265–1272. IEEE, 2018.

[58] Chen Liang, Jianbo Ye, Shuting Wang, Bart Pursel, and C Lee Giles. Investigating active learning for concept prerequisite learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[59] Liangming Pan, Chengjiang Li, Juanzi Li, and Jie Tang. Prerequisite relation learning for concepts in moocs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1447–1456, 2017.

[60] Zachary A Pardos, Steven Tang, Daniel Davis, and Christopher Vu Le. Enabling real-time adaptivity in moocs with a personalized next-step recommendation framework. In *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*, pages 23–32, 2017.

[61] Chidansh Bhatt, Matthew Cooper, and Jian Zhao. Seqsense: video recommendation using topic sequence mining. In *International Conference on Multimedia Modeling*, pages 252–263. Springer, 2018.

[62] Sudeshna Roy, Meghana Madhyastha, Sheril Lawrence, and Vaibhav Rajan. Inferring concept prerequisite relations from online educational resources. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9589–9594, 2019.

[63] Irene Li, Alexander R Fabbri, Robert R Tung, and Dragomir R Radev. What should i learn first: Introducing lecturebank for nlp education and prerequisite chain learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6674–6681, 2019.

[64] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[65] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[66] Jibing Gong, Shen Wang, Jinlong Wang, Wenzheng Feng, Hao Peng, Jie Tang, and Philip S Yu. Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 79–88, 2020.

[67] Wei Xu and Yuhan Zhou. Course video recommendation with multimodal information in online learning platforms: A deep learning framework. *British Journal of Educational Technology*, 51(5):1734–1747, 2020.

[68] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018, 1802.05365.

[69] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[70] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[71] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[72] Chris McCormick. Word2vec tutorial-the skip-gram model. *Apr-2016.[Online]. Available: http://mccormickml. com/2016/04/19/word2vec-tutorial-the-skip-gram-model*, 2016.

[73] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

[74] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. Specter: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, 2020.

[75] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. Pytorch-biggraph: A large-scale graph embedding system. *arXiv preprint arXiv:1903.12287*, 2019.

[76] Susan Thomas. Future ready learning: Reimagining the role of technology in education. 2016 national education technology plan. *Office of Educational Technology, US Department of Education*, 2016.

[77] Wenzheng Feng, Jie Tang, and Tracy Xiao Liu. Understanding dropouts in moocs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 517–524, 2019.

[78] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998, 2008.

[79] Shalini Pandey and George Karypis. Structured dictionary learning for energy disaggregation. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, pages 24–34, 2019.

[80] Sahil Gupta, Shalini Pandey, and KK Shukla. Comparison analysis of link prediction algorithms in social network. *International Journal of Computer Applications*, 111(16):27–29.

[81] Yiming Yang, Hanxiao Liu, Jaime Carbonell, and Wanli Ma. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 159–168, 2015.

[82] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun. Hierarchical reinforcement learning for course recommendation in moocs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 435–442, 2019.

[83] Chen Liang, Zhaohui Wu, Wenyi Huang, and C Lee Giles. Measuring prerequisite relations among concepts. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1668–1674, 2015.

[84] George D Kuh, Jillian Kinzie, John H Schuh, and Elizabeth J Whitt. *Student success in college: Creating conditions that matter*. John Wiley & Sons, 2011.

[85] John Self. Theoretical foundations for intelligent tutoring systems. *Journal of Artificial Intelligence in Education*, 1(4):3–14, 1990.

[86] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016.

[87] Mohammad Khajah, Robert V Lindsey, and Michael C Mozer. How deep is knowledge tracing? *arXiv preprint arXiv:1604.02416*, 2016.

[88] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. *CoRR*, abs/1808.09781, 2018.

[89] Tanja Käser, Severin Klingler, Alexander Gerhard Schwing, and Markus Gross. Beyond knowledge tracing: Modeling skill topologies with bayesian networks. In *International conference on intelligent tutoring systems*, pages 188–198. Springer, 2014.

[90] Penghe Chen, Yu Lu, Vincent W Zheng, and Yang Pian. Prerequisite-driven deep knowledge tracing. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 39–48. IEEE, 2018.

[91] Andrew S Lan, Andrew E Waters, Christoph Studer, and Richard G Baraniuk. Sparse factor analysis for learning and content analytics. *The Journal of Machine Learning Research*, 15(1):1959–2008, 2014.

[92] Radek Pelánek. Modeling students' memory for application in adaptive educational systems. *International Educational Data Mining Society*, 2015.

[93] Paul R Pintrich. The role of goal orientation in self-regulated learning. In *Handbook of self-regulation*, pages 451–502. Elsevier, 2000.

[94] Barry J Zimmerman. Self-regulated learning and academic achievement: An overview. *Educational psychologist*, 25(1):3–17, 1990.

[95] Jacquelynne S Eccles and Allan Wigfield. Motivational beliefs, values, and goals. *Annual review of psychology*, 53(1):109–132, 2002.

[96] Edward L Deci and Richard M Ryan. The" what" and" why" of goal pursuits: Human needs and the self-determination of behavior. *Psychological inquiry*, 11(4):227–268, 2000.

[97] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[98] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[99] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

[100] Lee Averell and Andrew Heathcote. The form of the forgetting curve and the fate of memories. *Journal of Mathematical Psychology*, 55(1):25–35, 2011.

[101] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[102] Haw-Shiuan Chang, Hwai-Jung Hsu, and Kuan-Ta Chen. Modeling exercise relationships in e-learning: A unified approach.

[103] Zhenya Huang, Qi Liu, Yuying Chen, Le Wu, Keli Xiao, Enhong Chen, Haiping Ma, and Guoping Hu. Learning or forgetting? a dynamic approach for tracking the knowledge proficiency of students. *ACM Trans. Inf. Syst.*, 38(2), February 2020.

[104] Youngduck Choi, Youngnam Lee, Dongmin Shin, Junghyun Cho, Seoyon Park, Seewoo Lee, Jineon Baek, Byungsoo Kim, and Youngjun Jang. Ednet: A large-scale hierarchical dataset in education. *arXiv*, pages arXiv–1912, 2019.

[105] Xu Wang, Diyi Yang, Miaomiao Wen, Kenneth Koedinger, and Carolyn P Rosé. Investigating how student's cognitive behavior in mooc discussion forums affect learning gains. *International Educational Data Mining Society*, 2015.

[106] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 495–503. ACM, 2017.

[107] Florent Garcin, Christos Dimitrakakis, and Boi Faltings. Personalized news recommendation with context trees. *arXiv preprint arXiv:1303.0665*, 2013.

[108] EL Deci and RM Ryan. (1985b). intrinsic motivation and self-determination in human behavior. new york: Plenum. 1985.

[109] John Sweller. Cognitive load theory. In *Psychology of learning and motivation*, volume 55, pages 37–76. Elsevier, 2011.

[110] Allan Collins and Richard Halverson. *Rethinking education in the age of technology: The digital revolution and schooling in America*. Teachers College Press, 2018.

[111] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[112] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. What to do next: Modeling user behaviors by time-lstm.

[113] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 46–54. ACM, 2018.

[114] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1269–1278, 2019.

[115] Aristides Gionis et al. Similarity search in high dimensions via hashing.

[116] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. Neural memory streaming recommender networks with adversarial training. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2467–2475, 2018.

[117] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. Embedding temporal network via neighborhood formation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2857–2866. ACM, 2018.

[118] Tomoharu Iwata, Amar Shah, and Zoubin Ghahramani. Discovering latent influence in online social activities via shared cascade poisson processes. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 266–274. ACM, 2013.

[119] Scott Linderman and Ryan Adams. Discovering latent network structure in point process data. In *International Conference on Machine Learning*, pages 1413–1421, 2014.

[120] Mehrdad Farajtabar, Safoora Yousefi, Long Q Tran, Le Song, and Hongyuan Zha. A continuous-time mutually-exciting point process framework for prioritizing events in social media. *arXiv preprint arXiv:1511.04145*, 2015.

[121] Mehrdad Farajtabar, Yichen Wang, Manuel Gomez-Rodriguez, Shuang Li, Hongyuan Zha, and Le Song. Coevolve: A joint point process model for information diffusion and network evolution. *The Journal of Machine Learning Research*, 18(1):1305–1353, 2017.

[122] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[123] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, pages 417–426. ACM, 2019.

[124] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 601–610, 2017.

[125] Xia Jing and Jie Tang. Guess you like: course recommendation in moocs. In *Proceedings of the International Conference on Web Intelligence*, pages 783–789, 2017.

[126] Jiezhong Qiu, Jie Tang, Tracy Xiao Liu, Jie Gong, Chenhui Zhang, Qian Zhang, and Yufei Xue. Modeling and predicting learning behavior in moocs. In *Proceedings of the ninth ACM international conference on web search and data mining*, pages 93–102, 2016.

[127] Yu Sun, Nicholas Jing Yuan, Xing Xie, Kieran McDonald, and Rui Zhang. Collaborative intent prediction with real-time contextual data. *ACM Transactions on Information Systems (TOIS)*, 35(4):1–33, 2017.

[128] Philip H Winne. Learning analytics for self-regulated learning. *Handbook of learning analytics*, pages 241–249, 2017.

[129] Alyssa Friend Wise. Learning analytics: Using data-informed decision-making to improve teaching and learning. In *Contemporary technologies in education*, pages 119–143. Springer, 2019.

[130] Alyssa Friend Wise. Designing pedagogical interventions to support student use of learning analytics. In *Proceedings of the fourth international conference on learning analytics and knowledge*, pages 203–211, 2014.

[131] Fareedah ALSaad, Assma Boughoula, Chase Geigle, Hari Sundaram, and ChengXiang Zhai. Mining mooc lecture transcripts to construct concept dependency graphs. *International Educational Data Mining Society*, 2018.

[132] Zenun Kastrati, Ali Shariq Imran, and Arianit Kurti. Integrating word embeddings and document topics with deep learning in a video classification framework. *Pattern Recognition Letters*, 128:85–92, 2019.

[133] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[134] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[135] He Bai, Peng Shi, Jimmy Lin, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. Segabert: Pre-training of segment-aware bert for language understanding. *arXiv preprint arXiv:2004.14996*, 2020.

[136] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*, 2017.

[137] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Aaai*, volume 14, pages 1112–1119. Citeseer, 2014.

[138] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[139] Yuanyuan Qiu, Hongzheng Li, Shen Li, Yingdi Jiang, Renfen Hu, and Lijiao Yang. Revisiting correlations between intrinsic and extrinsic evaluations of word embeddings. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 209–221. Springer, 2018.