




Providing personalized learning guidance in MOOCs by multi-source data analysis

Ming Zhang¹  · Jile Zhu¹ · Zhuo Wang¹ · Yunfan Chen¹

Received: 30 November 2017 / Revised: 22 March 2018 / Accepted: 1 April 2018 /

Published online: 8 May 2018

© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract Although millions of students have access to varieties of learning materials in Massive Open Online Courses (MOOCs), many of them feel lost or isolated in their learning experience. One of the potential reasons is the lack of interactions and guidance for individuals. Since MOOC students have diverse learning objectives, we propose to design different strategies for those students with different engagement styles. In this paper, we provide personalized learning guidance for MOOC students based on multi-source data analysis. We first conduct content analysis to identify key concepts in the courses. We then propose two structured model to evaluate student knowledge states by their quiz submissions. We also study on student learning behaviors and design a dropout prediction system. The experiments show the effectiveness of our algorithms and we discuss on the result both quantitatively and qualitatively. Last but not least, we employ a Web application of online student assessment service for both students and instructors, in order to display student learning states and provide suggestion for individuals.

This article belongs to the Topical Collection: *Special Issue on Social Media and Interactive Technologies*

Guest Editors: Timothy K. Shih, Lin Hui, Somchoke Ruengittinun, and Qing Li

✉ Ming Zhang
mzhang@net.pku.edu.cn

Jile Zhu
zhujile0918@pku.edu.cn

Zhuo Wang
wangzhuo@pku.edu.cn

Yunfan Chen
chenyunfan@pku.edu.cn

¹ School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China

Keywords Massive open online courses · Personalized guidance · Multi-source data analysis · Web application · Student assessment

1 Introduction

With the recent developments in online education, millions of students have benefited from Massive Open Online Courses (MOOCs). MOOCs provide a new way for life-long learning, which enable students to access abundant high-quality learning resources at their convenience and with no cost, such as watching video lectures, taking assignments and discussing on forums. Meanwhile, students with various learning objectives register at those MOOC platforms, like preparing for their future career, college life or just for fun. Despite all the advantages, MOOCs have suffered from extremely high rates of dropout. One of the potential reasons is that MOOC students are unavailable to personalized support and the lack of interaction between instructors and students would reduce their learning experience [9, 43]. One proposed solution is to provide MOOCs with personalized guidance through educational data mining techniques [44].

Personalized guidance in MOOCs is to provide MOOC learners with efficient learning resources and feedback according to their unique needs. Along with the development of platforms and the availability of the big data in MOOCs, more and more studies attempt to address the issue of personalization, such as early stage experimental results, system frameworks or system performance [44]. For example, some work studied on adaptive learning demonstrates that customizing content to the needs of individual learners have the potential to provide personalized learning [43]. Some previous explorations of course design suggested that personalized guidance would improve student learning experience and performance [3, 25]. Some work helped learners to reach the topics in which they are interested [49], and recommended learning materials based on confusion detection [1]. Recently, Pardos et al. proposed an adaptive intervention using a real-time data-driven recommendation framework [37], in order to increase learner navigational efficiency.

However, some recent attempts to address this issue also reported there is no evidence that providing interventions to learners was beneficial. For example, [21] recommended self-regulating learning strategies to MOOC students through a survey before their learning. Davis et al. [11] provided learners with retrieval cues to facilitate the active retrieval of information from memory. Tomkin and Charlevoix [45] conducted an A/B test on the course discussion forum, where the instructor was active in one forum while let students discuss among themselves in the other forum. We think one potential reason may be that such kinds of guidance were just designed for a small population of students who actively engage in courses.

For MOOC is accessible to everyone, students with different learning objectives will engage with different learning resources. As a previous study demonstrated, students who focus on learning have two fundamental activities, including viewing lectures and handing in assignments for credit [2]. The authors named students who primarily watch lectures the *Viewers* and named those who primarily hand in assignments the *Solvers*. Besides, for most courses, over 85% of students leave before they complete it [17]. Compared with those who want to finish the course, many of these students usually have relatively fewer interactions with the lectures and assignments, but spend relatively more time on forums, because they may get stuck and confused. These interactions are tracked by behavior logs. Therefore, in this paper, we consider the *Viewers*, the *Solvers* and the *At-risk* students as three main

types of engagement styles. However, since video lectures, assignment submissions and behavior logs are multi-source and heterogeneous, it is hard to design one single algorithm that effectively model all three kinds of students.

Intuitively, one of the solutions is to design multiple models for different types of students and then take appropriate interventions on targeted students. Thus, we first intend to identify the key concepts for those *Viewers* based on course content analysis. With such guidance, they can have an vision of key points before watching a lecture, or briefly review these concepts if they are going to take an assignment. We then consider to evaluate individual knowledge states for those *Solvers* by modeling quiz submissions. It can be useful for both instructors and learners, because it can remind students of their weakness and suggest instructors provide some explanations. We also try to discover *At-risk* students by observing their learning behaviors. The relationship between behavior patterns and dropout may help to improve course management and student retention.

Previous researches on MOOC extracted concepts and represented them as n-grams or clusters of questions with the help of external knowledge bases [33], manual annotation [15] or student quiz submissions [38]. Besides, Bayesian knowledge tracing is a classic model of student assessment [10], and recent studies improved it by deep neural networks [38] or explicit concept relations [18]. With regard to student behaviors, dropout prediction also attracted lots of attention in the past several years [26, 39, 48]. Different from these work, we leverage the structural characteristics of MOOCs to improve existing models.

In this paper, our main objective is to provide most MOOC students with personalized learning guidance. Thus, we propose a framework based on multi-source data analysis, which consists of content analysis, student assessment and dropout prediction, to satisfy students with different engagement styles. We first use a graph-based ranking method to extract concepts from video clips, which suggests students the most important concepts within the upcoming chapter. We then further develop two novel knowledge tracing methods, Multi-Grained-BKT and Historical-BKT, which model the structure of knowledge components and capture the correlations between submissions, to evaluate student knowledge states. We also explore the correlations between student behavioral data and dropout, and then design a system to predict student dropout based on history data before it happens. Experiments demonstrate the effectiveness of our algorithms. Especially, we design an online Web application to display student learning states and provide suggestion for individuals. To summarize, we have made following contributions:

- We propose a framework that provides personalized learning guidance for students with different engagement styles in MOOCs.
- We design several algorithms to resolve different situations and conduct extensive experiments to show the effectiveness.
- We employ an Web application of online student assessment service, which helps both students and instructors identify their individual knowledge states and overall performance.

The differences of this extended manuscript from our conference versions [8, 46, 52] mainly include:

- We propose a framework that leverages multiple sources of data to provide personalized learning guidance based on an Web application for visualization.
- We conduct further experiments to explain the effectiveness of our algorithms when predicting student performance.

- We introduce the procedure of data management which supports our online Web application.
- We design several Web interfaces to visualize student learning states and discuss the potential effects on both students and instructors.

The rest of the paper is organized as follows. Section 2 introduces the recent work related to the algorithms we proposed in this paper. After that, we describe the overview of the whole framework in Section 3. In Section 4, for we have described in the conference versions, we just briefly introduce the algorithms of our model, which leverage multiple sources of data for analysis. We also discuss on the experimental results quantitatively and qualitatively. Especially, we design a Web application and introduce its data management and interfaces in Section 5. At last, Section 6 concludes the paper and proposes several further researches.

2 Related work

In this section, we first briefly introduce recent MOOC studies on course design and interventions, which are the most commonly used methods to provide personalized learning guidance. Then, we respectively compare studies on concept extraction, knowledge tracing and dropout prediction, which are related with the algorithms we design in this paper.

2.1 Course design and interventions

One of the principal criticisms for MOOCs points to the lack of interaction. For most of students participate in MOOCs through the interaction with resources such as lectures, assignments and forums, many interventions were designed to enhance student engagement and improve their learning experience when they interact with these materials.

For example, previous work explored the effect of video production on student engagement [12, 23], which can be viewed as an instruction for teachers when preparing filmed lectures. Since confusion may lead to dropout, some work suggested detecting confusion in forums and sessions [7, 50], so as to provide just-in-time support for confused students. Besides, students tend to get confused if their assignments are graded without feedbacks, and evidence showed immediate feedback improves learning performance [24, 25]. And Basu et al. [3] presented an intervention that explicitly assists students in understanding detailed specification of technical assignments before their attempts.

Moreover, recommendation systems can be useful to assist students in identifying potentially helpful information objects [27]. Recent studies have explored the feasibility of setting up recommender systems on MOOC platforms to provide individual guidance. Jing et al. [16] developed and deployed a course recommendation algorithm based on multi-source data such as user interest, demographic profiles and course prerequisite relation. Khosravi et al. [19] used a collaborative filtering algorithm to recommend multiple-choice questions to individual students that address their interests and current knowledge gaps. Pardos et al. [37] proposed a real-time data-driven recommendation framework to predict next-step pages that students were likely to spend significant time on.

However, most of these work just designed the solutions to a specific task. Practically, MOOC students have different intentions when they enroll in a class, which can be reflected in their various learning traces and engagement styles [2]. For example, some of students primarily watch lectures, while some of them only hand in assignments for a grade. Unlike

these existing studies, we propose a novel framework that provides personalized learning guidance for majority of students. Specifically, this work utilize multi-source data to understand students with different potential demands and learning states.

2.2 Concept extraction in MOOCs

One of the general styles of engagement in MOOCs is watching video lectures, for it is a straight-forward way to learn. However, since students have different educational background, existing MOOC videos are not suitable for every student. Thus, one solution to this issue is to summarize the content of videos. As MOOCs are in fact unstructured knowledge bases, an intuitive and first-step idea is to extract concepts from textual content [15].

For example, some of recent work leveraged external knowledge bases to represent course concepts as n-gram terms and built prerequisite relations between them [32, 33]. Jiang et al. [15] proposed a semi-supervised method to annotate concepts appeared in the subtitles. Piech et al. [38] represented the concept as a cluster of problems by modeling sequential quiz submissions based on deep neural networks. However, the concepts extracted in these work have several limitations when transferring to a new specific MOOC, such as coarse-grained definition, manual labeling or great demand of quiz submissions.

Inspired by the work from Matsuda et al. [28], which applied Latent Dirichlet Allocation (LDA) model on assignment items and viewed the auto-generated clusters as knowledge component candidates, we transfer this method to the subtitles of MOOC videos. We then represent the concepts as vectors of word distribution in this paper. Considering that such auto-generated concepts may contain some redundant concepts, we then consider to filter meaningless ones by a graph-based ranking method. PageRank is a classic graph-based ranking algorithm and it is a common way to measure the relative importance of websites [31]. Some variants, like TextRank by Mihalcea, created an undirected graph from natural language texts for text processing, such as key-phrase extraction, extractive summarization [5, 29]. Different from these works, we propose an unsupervised algorithm for concept extraction, which leverages the structural information of document organization in MOOCs to capture the relation between concepts in different videos.

2.3 Knowledge tracing

Modeling student knowledge states is the key component to provide personalized learning guidance. MOOC students are available to well-designed assignments to consolidate knowledge and some of them even take repeated attempts for a higher grade. Knowledge tracing was proposed based on mastery learning where knowledge is composed of knowledge components. This concept was then introduced into intelligent tutoring system (ITS) [10] and modeled by dynamic Bayesian networks [41]. Bayesian knowledge tracing (BKT) model is now widely used, like Cognitive Tutor [42], to estimate whether one learner has acquired a specific knowledge component.

Classic BKT is provided with three assumptions: 1) There is no difference between students. 2) There is no difference between items for the same skill. 3) Students never forget what they have learned. A number of researches focused on how to extend the BKT. Pardos et al. [35] introduced the item difficulty to the model, called KT-IDEM, so that the model could estimate the difficulty of each item. Some work introduced student-specific parameters into BKT on a larger scale, which was powerful to predict the performance of unseen students [34, 51]. These work thought different students have different prior knowledge or

learning speed. All the extension above can make an improvement on predicting the correctness of next attempt. Moreover, Piech et al. [38] propose a Deep Knowledge Tracing model using deep neural network and obtain an obvious progress.

However, while BKT model is popular within Intelligence Tutoring System, there is only a little work researched on how to adapt this model to MOOCs. Pardos et al. [36] firstly introduced this model to MOOCs, but their definition of knowledge components and the use of behavior data are too simple and did not perform well. They regarded each question as a knowledge component, which ignored the fact that several questions in a quiz may be associated with the same knowledge component. Therefore, we considered to make the use of resource organization and submission sequence in MOOCs to improve the original model. We consequently propose two more effective models.

2.4 MOOC dropout prediction

In the era of MOOC, high dropout rate has drawn a lot of attention. Ramesh et al. [40] classified users into positive and negative ones, and models user participation to predict learning performances. The study in [6] used clustering method with data from three MOOCs to predict dropout. With the help of social network data and forum data, authors of [47] tried to predict the exactly dropout week with Bayesian based Temporal modeling approaches. A semi-supervised machine learning method of dropout prediction was proposed in [26].

Beside the research focused on dropout itself, some other recent researches paid attention to how to increase student engagement. In [48], the authors found that social engagement could promote commitment and therefore lower attrition. An comparison experiment performed in [30] revealed that whether there is support offered by a small group of tutor could influence the dropout. A survey and interview study in [20] suggested that restricting accessibility and limiting repeatability of online courses could help student persevere in a course. While our experiment and data statistics reveal a different suggestion. A neural network approach was proposed in [13] to improve MOOC efficiency by personalize learning resources for different students and hopefully to consequently reduce the dropout rate.

Compared to existing studies, our work is the first to dig into the explicit relationship between data tendency and dropout. We do not only investigate the data and dropout relation, but also develop a reasonable dropout prediction system based on the observation. Especially, our system is an unsupervised and online framework. We also discuss the data by different methods, including statistic analysis and a data mining experiment. We also make suggestions for course management based on the result of prediction system experiment and data observation. This is a complete investigation of student dropout and behavior pattern relationship.

3 Overview of proposed framework

As we mentioned, MOOC students enroll the classes with various intentions. Some of them spend most of time on watching videos, some of them try exercises again and again, and some of them explore the course to search the information they want. The main objective of our work is to provide MOOC students with personalized guidance, which potentially benefits student engagement and learning experience and plays an important role in further development of MOOCs. Therefore, in this paper, we propose a framework that leverages multiple sources of data to provide guidance satisfying students with different engagement

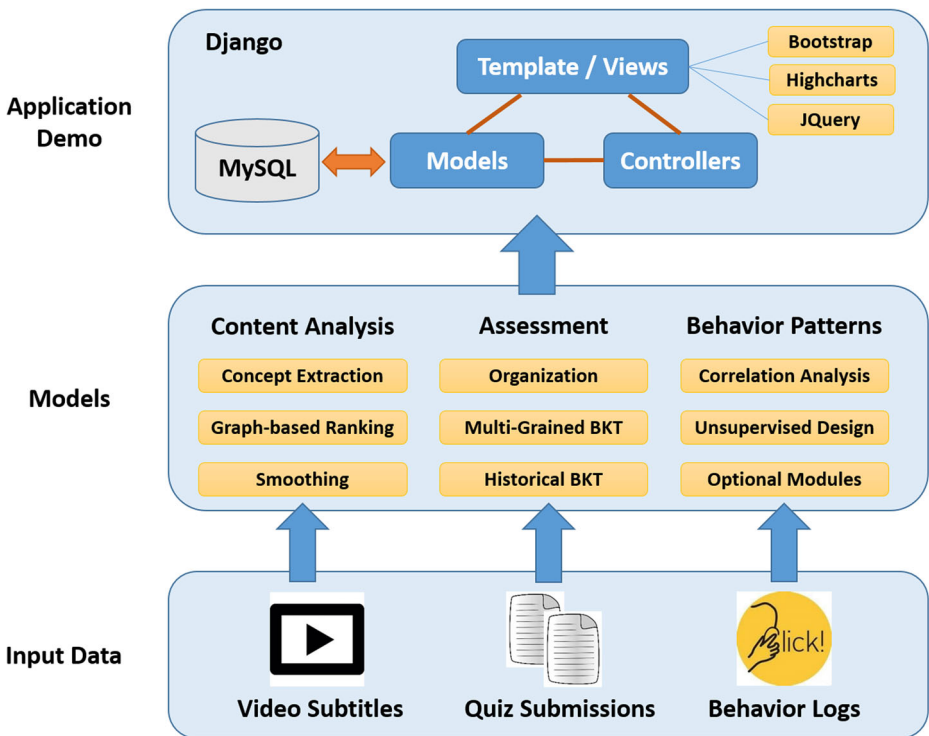


Figure 1 The proposed framework that leverages multiple sources of data to provide guidance satisfying students with different engagement styles. Data layer includes the model inputs such as video subtitles, quiz submissions and behavior logs. Model layer consists of multi-source data analysis like content analysis, student assessment and behavior patterns. Application layer shows a demonstration of our online Web service, which displays student learning states and provides suggestion for individuals

styles. The whole framework can be roughly divided into three main components, including Data, Models and Applications, as Figure 1 shown.

In the data layer, we consider that watching videos, taking exercises and browsing course materials are three principal interaction ways of MOOC learning. Therefore, we view subtitles as the text representation of video lectures to understand course content. We take student performance of quiz submissions to trace the knowledge states of individuals. We leverage log files, also called clickstream on Coursera, to capture student behavior patterns, which helps to identify at-risk students.

The model layer can be decomposed into three sub-modules that handle different data inputs respectively. We discuss the details of algorithms in Section 4. In brief, to understand what are the key concepts within a specific chapter, we extract latent concepts from the subtitles, and then take graph-based ranking to infer the most important concepts, accompanied by a smoothing strategy. To estimate the knowledge states of individuals, we first discuss the general knowledge organization of MOOC platforms and then design two algorithms, which leverage knowledge structure and sequential features in MOOCs. To associate student behavior patterns and their learning performance, we analyze the correlation between behaviors and dropout, and design an unsupervised online prediction system.

In the application layer, we design a Web application to show students their learning states, based on the analysis results generated from the model layer. We take Django¹ as our Web framework and use the MVC design mode. We also take MySQL² database as the data storage of our system. The visualization of data analysis is supported by several Javascript plug-ins, such as Bootstrap,³ Highcharts⁴ and JQuery.⁵ We describe our Web application and display some screenshots in Section 5.

4 Multi-source data analysis

4.1 Data preparation

The dataset for this paper consists of a Coursera course *Data Structures and Algorithms* from Peking University. The course lasts 14 weeks and consists of lecture videos, quizzes, programming assignments and a discussion forum. In total, there are 13,683 students who registered and visited the course web-pages. The course itself is in Chinese, along with English version of course materials and video subtitles.

Specifically, for the course, only 1,037 students still visited the course till the last three weeks and merely nearly a hundred students passed the course. Weekly quizzes consist of an average of 7.43 questions, which can be either multiple choice ones or blank filling ones. The questions in these quizzes are randomly sampled from a problem set, which contains 254 different items. We selected those who had submitted quizzes, and got a dataset of 1,077 students and 6,583 submissions. The labels that describe the relationship between questions and knowledge components were acquired from the teacher assistants.

To analyze the content of the lectures, we extract noun-phrases from each subtitle for preprocessing, based on a Python library. Previous studies demonstrated that nouns and noun-phrases tend to produce keywords that typically express what the content is about [1]. Thus, the lectures can be represented as lists of consecutive phrases. There are 3,964 different phrases in total, and each lecture has an average length of 129.4 (including repeated phrases).

4.2 Content analysis: identifying key concepts from video subtitles

In this section, we want to automatically tell students the most important concepts of the upcoming chapter, which can be helpful to those who focus on watching video lectures. Based on such guidance, they can have a vision of the course, or check whether they have achieved these concepts before they take an assignment. Figure 2 shows the overall architecture of our algorithm, which can be decomposed into three steps. In the first step, we use LDA model to generate concepts from the subtitles of lectures. In the second step, we define a particular PageRank method for ranking the importance of phrases. Finally, we apply transfer functions to reassign the importance value of phrases and measure the importance of concepts.

¹<https://www.djangoproject.com/>

²<https://www.mysql.com/>

³<http://getbootstrap.com/2.3.2/>

⁴<https://www.highcharts.cn/>

⁵<http://jquery.com>

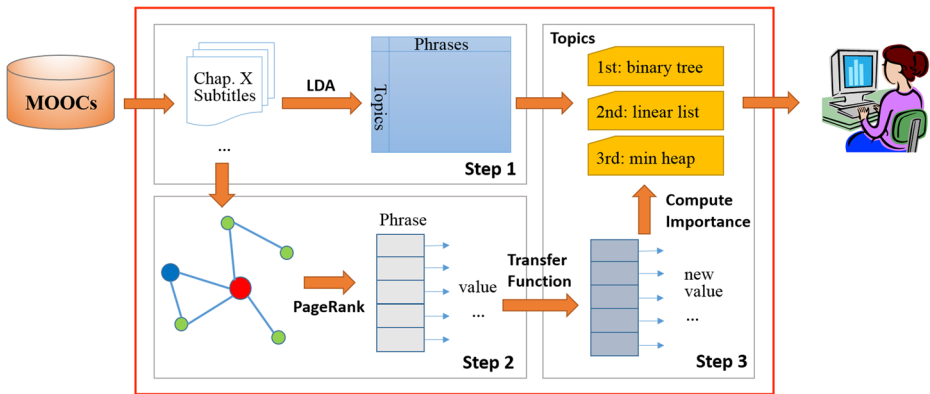


Figure 2 The procedure of the content analysis that takes subtitles of MOOCs as inputs and generates a ranked list of concepts to students

4.2.1 Generating concepts from subtitles

Inspired by previous work, which applied LDA model on assessment items [28], we transfer this method to the subtitles of videos in MOOCs. LDA model is a generative probabilistic model that allows a set of observations to be explained by unobserved groups [4]. In our cases, we consider concepts as topics, lectures as documents and phrases as words. Specifically, the model takes the phrase lists from a chapter as inputs, and returns a set of latent concepts, where each concept is characterized by a distribution over phrases. In addition, if the concepts have been predefined by experts (e.g. given n keywords for each concept), we can also take such information as an alternative, instead of generating by LDA model. For example, given the concepts in the form of n -gram, we just need to set the probabilities of corresponding phrases as $1/n$ and set the others as 0. Then, we can also obtain concepts in the form of phrase distribution.

The output of this step for each chapter is a set of latent concepts, in the form of probability distribution over phrases. This kind of concept representation is also interpretable to some extent. To have an intuitive sense, we display each concept as a tuple, including three phrases with the highest probability in the distribution. Table 1 shows the concepts generated from “Graph”, which is one of the chapters in this course.

Table 1 The concept candidates generated in Chapter “Graph”, each of which is represented as three phrases with the highest probability

Chapter 8: Graph
(Kruskal algorithm, algorithms, data structure)
(adjacency list, adjacent matrix, list contains)
(MST, Prim algorithm, minimum weight edge)
(DAG, start node, data structure)
(DFS, topological sort, post process)
(Dist, shortest path, source node)
(old value, time complexity, Dijkstra)

4.2.2 Ranking the importance of phrases

Our basic intuition is that important phrases are more likely to be mentioned in class. Moreover, when teachers talk about a new concept, they often briefly retrospect corresponding concepts as comparisons. Based on these latent relations, we design a particular PageRank method, which leverages both textual and structural information of lectures, to rank the importance of phrases within chapters. Our ranking algorithm can be decomposed into three processes, *Construction*, *Combination* and *Computation*. The output of this step is a ranked list of phrases, along with the value of their importance.

Construction PageRank is an algorithm for measuring the importance of website pages based on the web-graph [31]. In our cases, we denote the phrases as nodes and connect two phrases if they are close in the lecture. Formally, we define an undirected graph $G_k = (V_k, E_k)$ in the k^{th} chapter, where $V_k = \{v_1, v_2, \dots, v_{n_k}\}$ denotes the set of phrases. $L_k = \{l_1, l_2, \dots, l_{m_k}\}$ denotes the lectures in the k^{th} chapter. We follow the TextRank [29] to construct the basic phrase graph for each chapter and define the weight of edges as the times of co-occurrence between two phrases. The formula for the edge weight between phrases v_i and v_j is

$$w_k(v_i, v_j) = \sum_{s=1}^{m_k} \sum_{v_i \in l_s, v_j \in l_s} I \{dist(v_i, v_j) < \epsilon\}, \tag{1}$$

where I is an indicator function and $dist(v_i, v_j)$ denotes the offset difference between v_i and v_j . The formula implies that we take the co-occurrence of two phrases as the value of edge weight.

Combination For teachers usually avoid repeating concepts which have been discussed before, the relation of phrases will be insufficient if we only consider current chapter. In order to supplement more relationships in current phrase graph, we combine it with those of previous chapters. Therefore, we design a weighted method for the combination of graphs. Specifically, when we rank the phrases in a chapter, we combine the current phrase graph with those constructed by all previous chapters. We sum the weights of two phrases in different graphs by utilizing a damping factor α , which gives a lower weight to an earlier chapter. Formally, the weight of edges in the k^{th} chapter are formulated as

$$w_k(v_i, v_j) = \sum_{t=1}^k \alpha^{k-t} w_t(v_i, v_j). \tag{2}$$

Computation For each chapter, the output of this algorithm is a ranked list of phrases with the PageRank value. The PageRank value, transferred from a given node to the targets of its neighbors upon the next iteration, is divided by all adjacent nodes according to their edge weights. Formally, the iterative process can be described as the following equations. We first initialize all phrases with the same value as $PR_k(v_i; 0) = \frac{1}{N}$, where N is the total number of nodes. At each time step, the computation yields

$$PR_k(v_i; t + 1) = \frac{1 - d}{N} + d \sum_{v_j \in M(v_i)} \frac{PR_k(v_j; t) W_k(v_i, v_j)}{\sum_{v_s \in M(v_i)} W_k(v_i, v_s)}, \tag{3}$$

where $PR_k(v_i; t)$ denotes the PageRank value of v_i at time t in the k^{th} chapter, and $M(v_i)$ denotes the set of nodes adjacent to v_i .

4.2.3 Measuring the importance of concepts

By now, we have obtained the importance of each phrase. Note that the importance of same phrase does not necessarily remain unchanged in different chapters. However, PageRank method only concerns about relative importance and exaggerates the difference between top phrases. To avoid the situation where one phrase plays a dominant role in calculating the importance of concepts, we take three commonly-used distributions to smooth the result. The gradient will then be more gentle so as to alleviate the “slump” at first several phrase importance in the original ranking.

We multiply the phrase distribution of concepts and the vector of phrase importance. The product can be viewed as the importance magnitude of the concepts in this chapter. The formula is shown as:

$$Imp(Concept) = \sum_{phrase \in Concept} f_{trans}(Imp(phrase)) \cdot p(phrase), \quad (4)$$

where $p(phrase)$ denotes the probability of $phrase$ occurring in $Concept$ and f_{trans} denotes one of the transfer functions. At last, we sort the concepts by their importance, and output a ranked list of concepts as the final result within the chapter.

4.2.4 Experiments: ranking the importance of concepts within chapters

Setups We first generate concepts from the subtitles in each chapter. Then, we compute the importance of these concepts by our algorithm and get a ranking list. These concepts are also sorted by ground truth labels, which lead to an ideal ranking. Based on these two rankings, we then compute the metric score of our algorithm in this chapter. We take the average among chapters as the performance of our algorithm. To evaluate the performance of our algorithm, we consider four commonly-used strategies as baselines to rank the importance of phrases: (1) Random; (2) Bag-of-Words; (3) TF-IDF; (4) TextRank. For the comparability, these baselines also adopt the concepts generated from LDA model as ranking items.

Ground truth and metrics For students who want to complete the course are more likely to finish the quizzes and exams [2, 22], we think they pay a higher value on the concepts which count for more in the assignments. Thus, in this paper, we define the importance of a concept as “the number of problems that involve it”. Three domain experts independently annotated the relevance between the problems and the concepts. The Cohen’s Kappa for the annotations was 0.535 (in the range of $[-1, 1]$). This process induces a human-generated ranking, which is then compared to the ranking computed by our algorithm. We use three kinds of metrics to evaluate the effectiveness of our ranking algorithm: nDCG, MAP and Kendall’s τ , which are widely used for ranking model.

Quantitative results Table 2 shows the comparison of performance between baselines and our algorithms. We report seven variants of our algorithm, which differ in whether combines previous chapters as additional information and which transfer function is used for smoothing. We find the best variant (α -PR-Sigmoid) yields a 18.9 percent boost of MAP score. The results also show the consistency among different metrics. Besides, the methods which combine the content of previous chapters have a significant improvement,

Table 2 The comparison of performance between four baselines and our algorithm

Type	Algorithm	nDCG	MAP	τ_B
Baseline	Random	0.838	0.586	0.000
	BoW	0.867	0.631	0.007
	TF-IDF	0.850	0.580	-0.039
	TextRank	0.869	0.640	-0.010
Ours	PR-Linear	0.871	0.645	0.211
	PR-Sigmoid	0.883	0.649	0.256
	PR-Gaussian	0.878	0.613	0.144
	α -PR	0.900	0.749	0.263
	α -PR-Linear	0.920	0.752	0.237
	α -PR-Sigmoid	0.917	0.761	0.266
	α -PR-Gaussian	0.906	0.747	0.255

For all metrics, a higher value indicates a better performance

compared with those not combine. In addition, we find the transfer functions effective no matter whether or not the method combines the previous chapters.

We then discuss on the possible reasons why our algorithms perform better. Firstly, we think *PageRank method leverages the relation between phrases*. The PageRank method suggests that the phrase is important if the neighbors linked to it are important, so that an important phrase can be explored even if it does not occur so often. Then, *combining previous chapters provides the phrase graph with richer structure information*. One reliable explanation is that some phrases and relations not appearing in the current chapter play a role as “hubs” that connect two important phrases. At last, *transfer functions alleviate the bias from PageRank*. For the importance of top phrases have been exaggerated in PageRank, the concepts having these phrases with a higher probability will dominate the others.

Hyper-parameter sensitivity When we combine the graphs of previous chapters, the damping factor α should be preset. Note that when α equals to 0, the method will degrade into those not combining the previous chapters. We find that when α tuned from 0.05 to 1.00, the performance of variants using transfer functions gets worse, while the performance of variants using PageRank value directly remains unchanged in most of the time but has an increase at 1.00. Therefore, we set α to 0.05 if we use a transfer function for smoothing and set it to 1.00 otherwise. The situations are consistent when using different metrics. We think it because when using a transfer function, a lower value of α enables the current graph to enrich the structure information without influencing the relation between phrases. However, when using the original value, the importance of top phrases were exaggerated, so that α was set as 1.00 to “dilute” the effect of top phrases.

Case study The experiments have shown the performance of ranking the concept importance within chapters, which is useful for students to know the emphasis of upcoming lectures. Moreover, when students prepare for exams, our framework can also guide students according to their learning status. We assume that two students (S_A and S_B) are preparing for the mid-term exam, including 8 chapters. S_A learned all the content well, while S_B is deficient in “Linear List”, “Queue and Stack”, “Binary Tree Application” and “Tree and Forest”. We take all subtitles as inputs for S_A , so that we can design an overall review plan. While we just take subtitles in those four chapters as inputs for S_B , in order to concentrate on the concepts of weak points. The results are shown in Table 3.

Table 3 The top five concepts suggested in two cases

	Rank	Concepts for S_A	Concepts for S_B
Each concept is concluded with one phrase here	1	Logical structure	Sequential list
	2	Complete binary tree	Linear list
	3	Linear list	binary search tree
	4	Binary tree structure	Binary tree structure
	5	Binary tree traversal	Tree structure

$Case_A$ shows that our algorithm suggests the concept “binary tree” as most important content. In fact, the tree structure is indeed the most important in the first half of the course, for three chapters introduce the foundation, application, and extension of binary tree separately. In $Case_B$, our algorithm puts more emphasis on “linear list”. One reliable explanation is that linear list is a fundamental data structure and the instructor frequently mentions it when introducing the implementations of queue, stack, tree structure.

4.3 Assessment: tracing knowledge states by structured models

Bayesian Knowledge Tracing (BKT) is widely used in intelligent tutoring system (ITS), which models each learner’s mastery of knowledge components. Such a student assessment model can be useful for both instructors and learners, for it can infer the weakness of students and then suggest instructors provide further explanations. However, it is quite different between MOOC and ITS when modeling students by BKT. For example, all the questions appeared in ITS are associated to one or more specific knowledge components, which were labeled by domain experts beforehand. Thus, it is more convenient to build question sequences for the same knowledge component in ITS. On the other hand, MOOC has its own advantages such as the hierarchical organization of video clips, which is helpful to model optimization. Therefore, making full use of the characteristics of MOOC plays an important role in MOOC student modeling.

4.3.1 Knowledge organization in MOOCs

Classic BKT models student knowledge as a latent variable and updates by observing the correctness of each student’s interactions with the questions involving the knowledge. Therefore, in order to trace student knowledge states using BKT in MOOCs, we should first define what interactions and knowledge components are. Notice that most MOOC platforms allow multiple submissions for a quiz, so that each submission can be regarded as an update of knowledge states. Moreover, MOOC platforms, like Coursera, allow several variations for the same question, so that questions are similar but different between sequential trials for the same quiz.

However, compared to the mastering learning context, there is usually no explicit knowledge component definition on MOOC platforms. Considering that each chapter talks about a specific topic, a straight-forward solution is to regard a whole chapter as a knowledge component. We define the BKT model based on this definition as *Coarse-BKT*, which is one of our baselines. However, such a knowledge component definition is too coarse and it is hard to model student knowledge states precisely.

An alternative way which enables fine-grained knowledge tracing is to leverage the structural information of video clips in MOOCs. Our intuition is that the knowledge components in MOOCs are usually hierarchically organized, which means each chapter consists of several lecture videos and each video usually focuses on a specific topic. We can thus consider the content covered by each video as a knowledge component. We denote the baseline, which applies the BKT on the fine-grained knowledge component definition, as *Fine-BKT*.

Such definition of knowledge component has several advantages:

1. Because of the video organization in MOOCs, this kind of definition can be easily generalized to most courses.
2. MOOCs usually provide abundant materials for lecture videos, e.g., slides and in-video quizzes. This makes knowledge components well-defined and interpretable, as compared to clustering-based methods [28].
3. Computers can handle knowledge components through natural language processing techniques automatically by using texts like subtitles and forum posts.

However, the quizzes on MOOCs are usually designed for a whole chapter, so that we should figure out which video each question is associated with. Then, we can update knowledge states through corresponding questions. In this paper, we asked two teaching assistants who designed the quizzes to label the relations between video clips and questions.

4.3.2 Multi-grained-BKT

Intuitively, we think that the knowledge components within a chapter tend to be related to each other, while Fine-BKT models each knowledge component respectively. To improve the performance of Fine-BKT, we propose a novel model called *Multi-Grained-BKT*, which address the issue above. In this model, we still regard the whole chapter as an overall coarse-grained knowledge component, but regard the content covered by each video clip as one of its fine-grained knowledge components. Specifically, when knowledge components within a chapter are independent, the possibility of mastering fine-grained knowledge plays a more important role on the model, while the possibility of mastering coarse-grained knowledge works if the knowledge components are dependent.

Figure 3 shows that Multi-Grained-BKT has an additional layer of “fine-grained knowledge components”. When a student masters a coarse-grained knowledge component, there

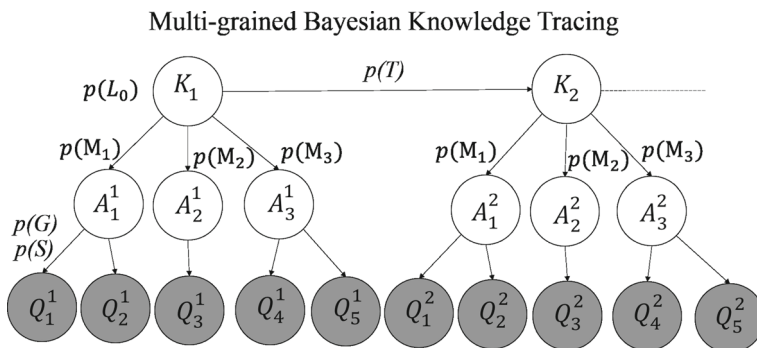


Figure 3 Multi-Grained-BKT: there are multiple fine-grained knowledge components for a coarse-grained knowledge component with one or more questions. Node A_k^i represents the knowledge state of fine-grained knowledge component k at time t . $p(M_k)$ represents the probability of mastering fine-grained knowledge component k when the overall knowledge component has been mastered

is a probability of mastering one of its fine-grained knowledge components, modeled by $p(M_k)$. Otherwise, he/she cannot master any of its fine-grained knowledge components, i.e.,

$$\begin{aligned} p(A_k^t = 1) &= p(L_t)p(M_k) \\ p(A_k^t = 0) &= p(L_t)(1 - p(M_k)) + (1 - p(L_t)). \end{aligned} \tag{5}$$

The parameter $p(M_k)$ models the difficulty of mastering a specific fine-grained knowledge component k . We pre-specify its value as the average correct rate of relevant questions:

$$p(M_k) = \frac{\#\{\text{correct responses for } k\}}{\#\{\text{all responses for } k\}} \tag{6}$$

We thus control the model complexity to four free parameters as before, and use EM algorithm to estimate the parameters and predict quiz submission results.

4.3.3 Historical-BKT

As mentioned, each submission is regarded as an update of knowledge states, and MOOC platforms allow variations for a question in sequential trials. These variations are closely related, usually about the same problem with different operational data or different multiple choice options. To make use of this characteristic, we propose a novel method called *Historical-BKT*, as shown in Figure 4. The probability of guessing and slipping will depend on the previous question response. Intuitively, if previous response is correct, $p(G)$ tends to be larger and $p(S)$ tends to be smaller. Therefore, we formulate this model by following equations:

$$\begin{aligned} p(Q_k^t = 1) &= p(L_t)(1 - p(S|Q_k^{t-1})) + (1 - p(L_t))p(G|Q_k^{t-1}) \\ p(Q_k^t = 0) &= p(L_t)p(S|Q_k^{t-1}) + (1 - p(L_t))(1 - p(G|Q_k^{t-1})) \end{aligned} \tag{7}$$

In addition, for there are three different states for responses in the previous submission (i.e., Correct, Incorrect, and Not existing), this model has eight parameters in total.

4.3.4 Experiments: predicting correctness of question responses

Setups To evaluate the performance of our algorithms described in Section 4.3, we compare all four models on predicting student question responses in the last submission with

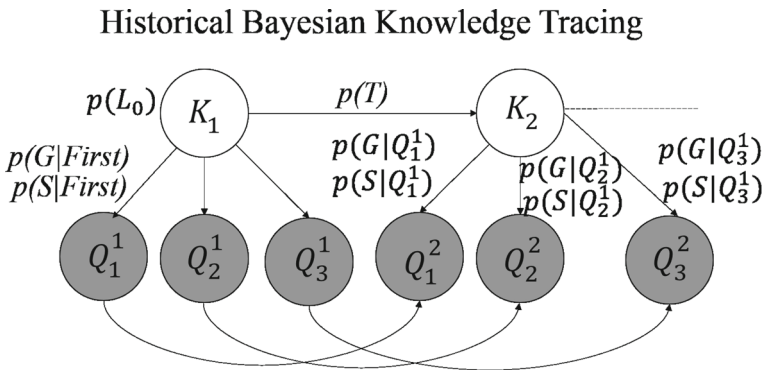


Figure 4 Historical-BKT: The response depends not only on the knowledge component, but also on the previous question response. That is, $p(G)$ and $p(S)$ of node Q_k^t depend on node Q_k^{t-1}

a 5-fold cross-validation [36]. Specifically, for each weekly quiz, we take the student submissions of training set to infer the parameters in our model. And then, for each student, we predict the performance of last submission based on his or her previous submissions in the testing set. We use the area under the curve (AUC) as the metric, which is widely-used in binary classification. One of its advantages is the robustness of validity when the dataset is unbalanced. We enumerate all pairings of correct and incorrect responses and the AUC is simply the percentage of the pairings where the correct responses get higher predictive probability over the incorrect responses. This metric ranges in $[0, 1]$, and it reaches at 1 when the model performs the best.

Quantitative results Figure 5 shows the AUC results of the four models on all the 14 quizzes, and Table 4 lists the two-tailed paired t-tests to evaluate the significance of the performance difference. The AUC of Fine-BKT is not significantly different from that of Coarse-BKT, showing that only using the fine-grained knowledge component is not sufficient to model students knowledge states. In contrast, Multi-Grained-BKT achieves a significant improvement of 0.0351 over Coarse-BKT (p -value=0.0015). The reason is that Multi-Grained-BKT models the hierarchical structure of knowledge components in different granularities, and is able to capture the relatedness between fine-grained knowledge components in the same chapter. Historical-BKT, which captures the relations between multiple quiz submissions, performs as well as the Multi-Grained-BKT (p -value=0.1801) and achieves a significant improvement of 0.0462 over Coarse-BKT (p -value=0.0003).

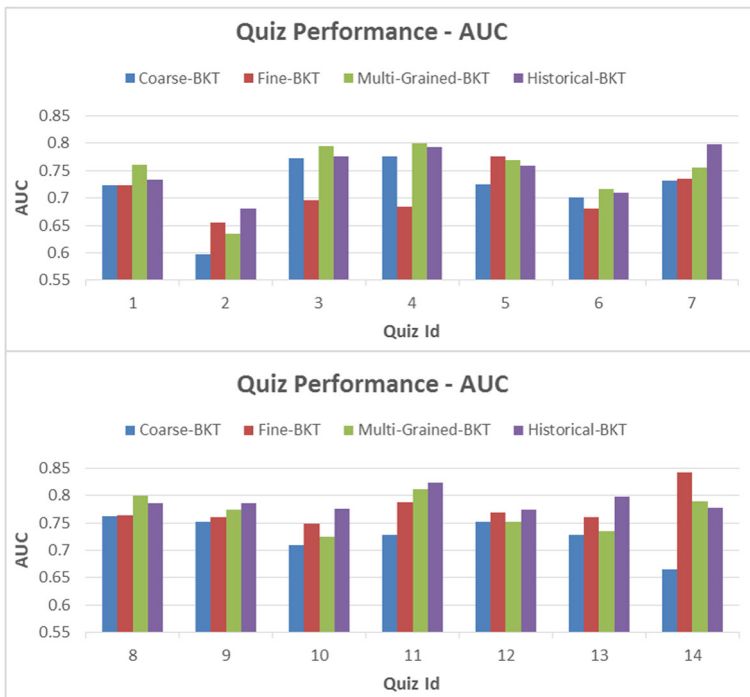


Figure 5 Cross-validated AUC results of the four models on all 14 quizzes. Multi-Grained-BKT and Historical-BKT achieve higher AUC in most quizzes

Table 4 The average AUC results on all 14 quizzes and the p-values of t-tests between Coarse-BKT and the other three, which shows the significance of the improvement

Model	Coarse-BKT	Fine-BKT	Multi-Grained-BKT	Historical-BKT
Average AUC	0.7233	0.7415	0.7584	0.7695
p-value	–	0.3110	0.0015	0.0003

Parameter analysis The reason why Historical-BKT performs the best is related to the construction of model and the course design. As we view each submission as a time-stamp, the same question appeared in different submissions are extremely similar to each other, for we just change the data used for calculation. Thus, students are more likely to response correctly if their answer was already correct in the previous submission. In fact, we investigate a statistical result of on the first five quizzes, as shown in Table 5, where *C2C*, referring to *Correct to Correct*, denotes the situation where students answer the same question correctly in two consecutive submissions, the same as *Wrong to Wrong*. We find that given a question, a student who answers correctly in previous submission can also give a correct answer in this submission nine times out of ten. Due to such a significant relation between two submissions, the design of Historical-BKT works in this dataset. Table 6 shows the parameters of model trained by submissions of the first quiz. From the result, we find that when a student is in the knowledge state of unlearned and guess the answer correctly, it is more likely that he or she also guess correctly in the last submission. Besides, we also find that when a student in the knowledge state of learned but accidentally submit a wrong answer, it is also more likely that he or she give a wrong answer last time. Therefore, we demonstrate that there do exist strong relations between results of two adjacent submissions. That also partially explains the reason that Historical-BKT performs well.

4.4 Behavior patterns: understanding how at-risk students engage

MOOCs have suffered from extremely high rates of dropout. For most courses, over 85% of students leave before they complete it [17]. Therefore, an effective dropout prediction system will be helpful to discover at-risk students and take appropriate interventions. In this section, we consider the students as the dropouts if they remain inactive for more than two weeks consecutively until the very last week. We first conduct a statistical study on student behaviors to analyze the factors which may cause dropout. The result could help us design the prediction system. Then, we present the general design of our system, including the basic components and optional components. We finally conduct experiments on two courses

Table 5 Statistics of submissions in the first five quizzes

Chapter	C2C	C2W	W2W	W2C	Ratio of “C2C”	Ratio of “W2W”
1	2315	87	523	873	0.964	0.375
2	2371	84	573	926	0.966	0.382
3	2355	94	544	913	0.962	0.373
4	2275	94	537	898	0.969	0.374
5	2372	85	563	934	0.965	0.376

“C” refers to *Correct* and “W” refers to *Wrong*

Table 6 Parameter analysis for Historical-BKT shows that $p(G|PreviousCorrect) > p(G|PreviousWrong)$ and $p(S|PreviousCorrect) < p(S|PreviousWrong)$

	Previous wrong	Previous correct	First submission
$p(G)$	0.3099	0.5687	0.5178
$p(S)$	0.3211	0.0578	0.1683

The result indicates that a student is very likely to response correctly if he/she is correct in the previous submission

and discuss on the results. In order to reasonably label the data when course is in progress, we study the relationship between inactive time period and final dropout. We find that if a student is inactive for more than 3 weeks, the student will be more likely to leave the course. Thus, we label the students with the facts that whether there is any activities in subsequent three weeks. We test our system from week 4 to the end of course in Section 4.4.3 based on this labeling strategy.

4.4.1 Statistical study on behavioral data

We first conduct a correlation analysis on the behavior data to check whether these patterns are correlated with dropout. Besides, we also take the difference between weeks into account. The results are shown in Table 7. We then draw following guidelines for course design.

1. **Course Attendance:** A visiting of the course website can help students keep up with the course progress. A weekly reminder may be helpful for students.
2. **#Web-page Viewed:** To encourage students view more course materials, a reminder consists of a table of contents for material would be helpful.
3. **Video Watching:** To encourage students to watch videos, the links to the videos could be included in the weekly reminder.
4. **#Video Pause:** For online course, students may hardly keep focusing on the course material for a long period of time. So we could design some in-video quizzes to help to divide the video into short pieces.
5. **#Quiz & #Assignment Attempts:** We can help student follow the course by allowing students to try multiple times for an assignment or a quiz.
6. **Forum:** Almost all types of forum participation could make a student less likely to drop out. It should be a good strategy for course managers to encourage students to participate in the forum discussion.

From the result, we can see that the difference between weeks does not correlated with dropout. This is because that *the 14 weeks of the course are independent from each other compared with other courses*. This observation is helpful for us to design the prediction system. Therefore, we make these features as optional in our system, that is, we basically use the 15 features present in Table 7 except for the features that start with Δ .

4.4.2 System design

Our system is an online algorithm running on weekly behaviors. To get the feature vector for each student, our system samples the data once a week and processes with a subtractor

Table 7 The correlation analysis on student behavior patterns and dropout

Category	Data	$r(\text{Correlation})$
View	Visit Course (True or False)	0.300**
	#Pages Viewed	0.390**
	$\Delta\#\text{Pages Viewed}$	-0.002
Learn	#Videos Watched	0.146**
	$\Delta\#\text{Videos Watched}$	0.149**
	Ave. #Pauses while Watching	0.149**
	Playback Speed	0.035
Test	#Attempts on Quiz	0.198**
	$\Delta\#\text{Attempts on Quiz}$	0.021
	#Attempts on Programming	0.357**
	$\Delta\#\text{Attempts on Programming}$	0.013
Discuss	#Visits of Forum	0.248**
	$\Delta\#\text{Visits of Forum}$	-0.063*
	#Posts Viewed	0.245**
	$\Delta\#\text{Posts Viewed}$	-0.039
	#Posts Posted	0.161**
	$\Delta\#\text{Posts Posted}$	0.031
	#Comments Posted	0.106**
	$\Delta\#\text{Comments Posted}$	0.009
	#Labels Added	0.042
	$\Delta\#\text{Labels Added}$	0
	#Labels Deleted	b
$\Delta\#\text{Labels Deleted}$	b	
#Likes	0.083**	
$\Delta\#\text{Likes}$	0.006	
#Dislikes	0.042	
$\Delta\#\text{Dislikes}$	0.042	

*: significance at the $\alpha = 0.05$;
 **: significance at the $\alpha = 0.01$;
 b : too less data to analyze correlation;
 #: number of;
 Δ : change of value between two consecutive weeks.

and an attenuator before classification. The work-flow for week n is shown in Figure 6. We discuss the effectiveness and how to select the optional components in Section 4.4.3. The classifier can be any kind of classification method which takes feature vector as an input and finally outputs a binary result. In this paper, we use the Random Forest [14] to predict dropout.

Since our discussion in Section 4.4.1 reveals the fact that the difference of student behaviors is not correlated with whether a student will dropout, we make the subtractor as an optional module in our system. Actually, according to the courses which need students qualified with the knowledge of previous weeks, the subtractor will play an important role in improving system performance. We concatenate the subtractor output and the original feature vector as the new representation of students. Formally, we define the new feature vector Y_n as

$$Y_n = \begin{pmatrix} X_n \\ \nabla X_n \end{pmatrix} = \begin{pmatrix} X_n \\ X_n - X_{n-1} \end{pmatrix} \tag{8}$$

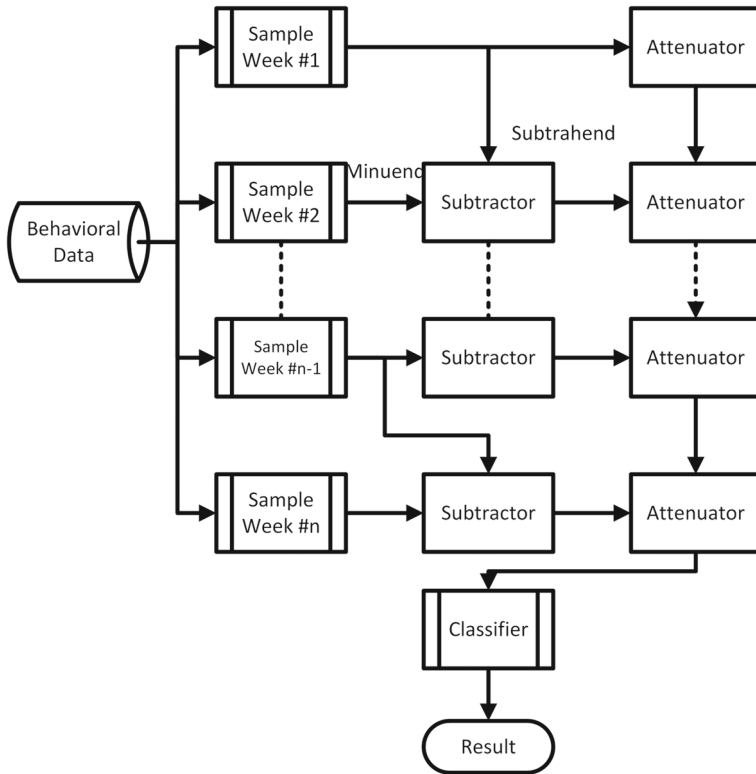


Figure 6 The work-flow of our dropout prediction system, which contains a basic feature extractor and two optional components

If we choose not to use the subtractor, $Y_n \equiv X_n$.

The attenuator in our system is designed based on the fact that recent behaviors tell more about whether a student will dropout in following weeks. We choose exponential decay for our attenuator. The decay constant λ satisfies $0 \leq \lambda \leq 0.5$. We get a feature vector Z_n output from the attenuator as

$$\begin{aligned}
 Z_n &= \lambda^{n-1}Y_1 + \sum_{k=2}^n (1 - \lambda)\lambda^{n-k}Y_k \\
 &= \begin{cases} Y_n & \text{if } n = 1 \\ \lambda Z_{n-1} + (1 - \lambda)Y_n & \text{else} \end{cases} \tag{9}
 \end{aligned}$$

The larger λ is, the more historical information remains. If we set $\lambda = 0$, it is exactly the system without the attenuator. If we choose to not use the subtractor, $Z_n \equiv Y_n$.

As shown in Section 4.4.3, for those strongly content related courses, we should choose subtractor but not attenuator. For weakly content related courses, the attenuator could improve the accuracy of our system but not subtractor.

4.4.3 Experiments: predicting MOOC student dropout

In this section, we first conduct our experiment on course *Data Structures and Algorithms* as described in section 4.1. Then, we discuss the result in detail to better understand the

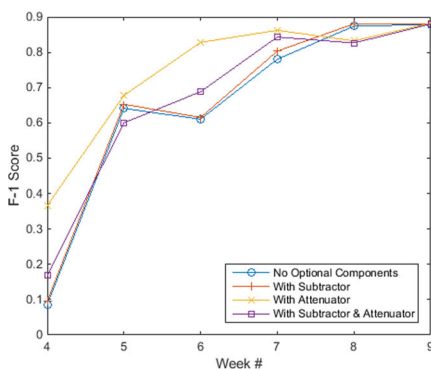
system and offer some suggestions on course improvement. We also select another course *Introduction to Computing* by Peking University on Coursera to examine whether our system works on other courses as well. During the following experiments, we set $\lambda = 0.5$ for the attenuator because of the performance reported in [8].

There are four different settings for our dropout prediction system – No Optional Components, With Subtractor, With Attenuator, and With All Optional Components. We compared the system F-1 Score under four settings and the results are shown in Figure 7a. When $n \leq 7$, the system with attenuator performs the best. When $n \geq 8$, four settings have similar performance and the system with subtractor has the best performance. We partially ascribe it to those students who registered for the course but did not intend to finish the course. These students will visit the course material frequently in the first few weeks and cause a lot of noises. Therefore, the system with attenuator, which reduces such noise by aggregating the behaviors of several weeks, predicts the dropout better.

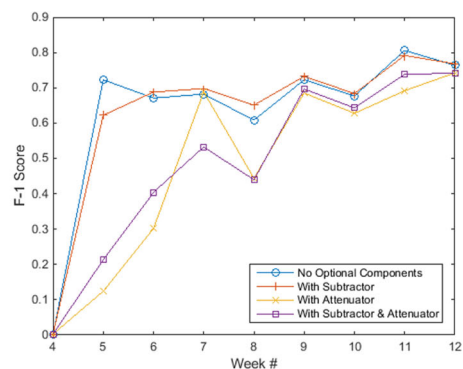
On the other hand, subtractor performs not well in *Data Structures and Algorithms* course, especially in the first seven weeks. We think it because that the course introduces a specific data structure weekly in the first half of the course. The low relevance between course materials of different weeks makes it possible to learn the course in any order. That is, we cannot tell whether a student will dropout simply by the fact that the student visit the course less than before. However, from week 7, the subtractor has a relatively better performance. This is because the course contents require knowledge introduced in previous weeks from week 7 (graph, sort, index, etc.). Thus, we call such a kind of courses as weakly content related courses. Temporal decrease of interactions will not lead to the dropout in a weakly content related course.

We also conduct the same experiment on course *Introduction to Computing*. This course consists of basic computer principles and basic knowledge of C++ programming. It lasts for 12 weeks and also has quizzes, programming assignments, examinations and discussion forum. In brief, this course has the same setting as the course *Data Structures and Algorithms* but lasts for more weeks. The experiment results are shown in Figure 7b.

The system with subtractor has the best performance among all four settings. We see that the system with attenuator performs worse in *Introduction to Computing*. We think the reason may be the highly related content, that is, the content in this course requires the knowledge introduced in previous weeks to understand. Therefore, analyzing course characteristic in advance can help us choose proper system components. We conclude that



(a) Data Structures and Algorithms



(b) Introduction to Computing

Figure 7 The performance of dropout prediction on two Coursera courses

if a course is highly content related, the absence will more likely to invoke a student dropout compared with a weakly content related course.

In addition, we also compare the system performance between two courses. We find that the system on *Data Structures and Algorithms* performs better, which may be caused by the fact that *Introduction to Computing* was open for registration during the whole course period while *Data Structures and Algorithms* did not accept new students after the first two weeks. Besides, for whether the learning materials of different weeks are related with those of previous weeks matters on dropout, we can help students manage their study by indicating dependencies between course materials.

5 Application system

In this section, we introduce our Web application system of online student assessment. Firstly, we describe the procedure of data management in our system, including how we process text data collected from MOOC providers and store structured content into databases. Then, we illustrate the details of online module, such as Web framework and the tools used for online visualization. At last, we display several Web interface screenshots, and explain how can our Web application work on providing guidance for MOOC students.

5.1 Data management

Given the data collected from MOOC providers, we first extract text content, like subtitles and question content, and carry out a data cleaning process to improve the data quality, such as Chinese word segmentation and removing stop words. After that, in order to automatically label questions with knowledge components, we calculate the TF-IDF (term frequency – inverse document frequency) as the representations of questions and knowledge components. Note that in this paper, we define a knowledge component as a piece of video clip, as described in Section 4.3. Based on such representation, We can measure the similarity of a specific question and knowledge component, using the cosine distance as the measurement, and then label the questions with the closest knowledge components. We finally store this kind of relationship as a Q-matrix, which would be used in our knowledge tracing algorithm. Figure 8 shows the procedure of data management.

Specifically, since we define knowledge components as the video clips in MOOCs which are separated by weeks, we can then take the subtitles of video clips as the natural language representation of knowledge components. In the first step, we leverage a Python Chinese text segmentation module to process text content of subtitles and questions exported from MOOC database. We also take both a general stop word list and a domain-specific stop word list to remove irrelevant words, so as to obtain a more accurate representation of knowledge.

In the second step, we calculate TF-IDF representations for subtitles and questions, which reflect how important a word is to a document given the text collection. We list the formulation of the variant we used during our implementation as follows, where D denotes the document set, N denotes the total number of documents in the corpus, and t denotes a term in a specific document d :

$$tf(t, d) = f_{t,d} / \sum_{t' \in d} f_{t',d}$$

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

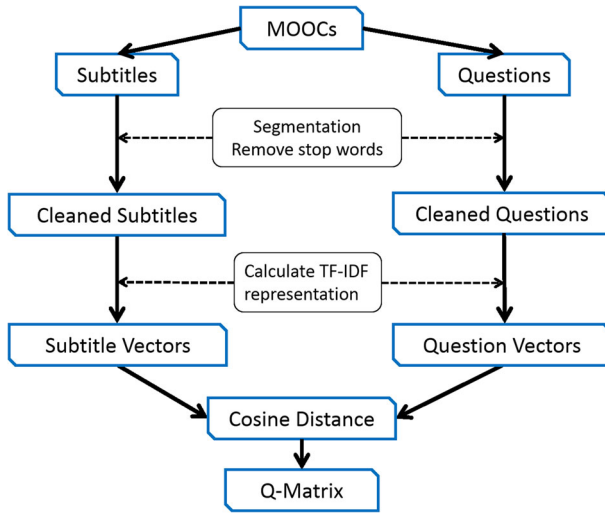


Figure 8 The procedure of data management, including text content extraction, segmentation, tf-idf representation and Q-matrix annotation

Therefore, we can represent subtitles and questions as vectors of term importance. Table 8 displays two examples of $TF - IDF$ representation.

The last step is to label the questions with knowledge components, i.e. subtitles of video clips. Intuitively, we think a question contains a specific knowledge component if their semantic representation are similar. During our implementation of this system, we simply use cosine distance to measure the similarity between two kinds of text content. To simplify the model, we label the question with the most similar knowledge component. As we have the manually labeled result as the ground truth, mentioned in Section 4.3, we compare it with this automatic labeling to evaluate the effectiveness of our system. The average precision is 75.8%, and the precision of each quiz is shown in Figure 9. From the result, we find that automatic labeling method performs well on most quizzes, however, it is unsatisfactory

Table 8 The tf-idf representation of two video clips

String and Brute-Force		KMP Fast String Matching	
Keywords	tf-idf	Keywords	tf-idf
Substring	0.475	Pattern array	0.320
String	0.266	Next	0.264
Storage	0.247	Equal	0.251
Character	0.241	Algorithm	0.251
Position	0.228	String	0.226
Pattern	0.215	Fact	0.201
Inside	0.203	p(k)	0.188
Pattern matching	0.190	Beginning and end	0.188
Target	0.177	Matched	0.176
Method	0.152	Situation	0.176

We list top 10 words with the highest value and translate into English from Chinese

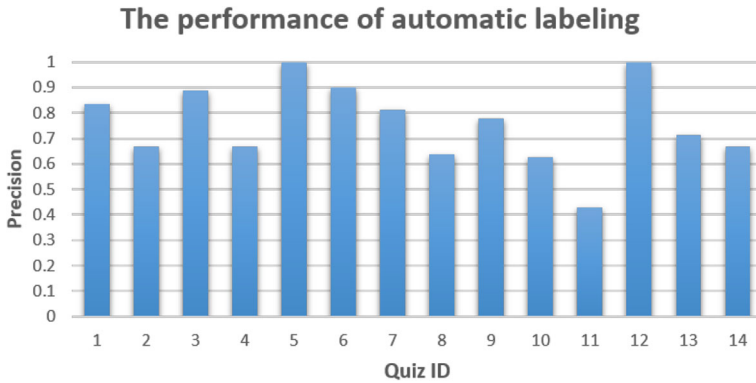


Figure 9 The performance of automatic labeling among 14 quizzes in this course

when the questions involves multiple knowledge components. For example, the 11th Quiz contains the knowledge of Chapter “Retrieval”, and most questions in this quiz require students to understand the concepts of retrieval, hash table and hash function (there exist lots of proper nouns in the questions), which reduces the performance of labeling in this system.

5.2 Web application framework

In general, we use Django, which is a high-level Python Web framework that encourages rapid development and clean pragmatic design, to build up our system. Besides, we employ model–view–controller (MVC) framework as the architectural pattern for implementing our user interfaces on the websites. Considering that our algorithms are off-line models with the aim of avoiding delay and redundant calculation, we need to store the results into the database beforehand. Thus, we take MySQL, which is the most popular open source database, to manage and provide data for our Web application. To design the websites, we choose Bootstrap as our front-end Web framework, for it contains various basic templates for typography, forms, buttons, navigation and other interface components. We also use Highcharts and JQuery as optional JavaScript extensions to enable data visualization and data analysis. The whole framework is organized as Figure 10.

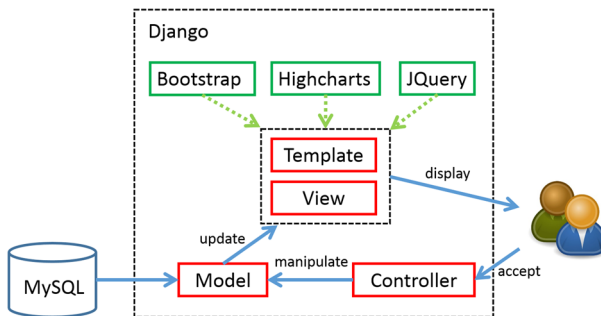


Figure 10 The architecture of our Web application framework

For example, when users turn up on our Web application, the controller will receive the requests and then send commands to the model. The model then calls the view to display the homepage of application system. After that, the user click a hyperlink on the page to search specific information, and the controller will send a new command to the model, including a URL with several parameters or SQL queries. The model then accesses data from MySQL database according to the parsed command and sends the result to the view. The view finally generates new layout to the user based on data given by the model.

5.3 Interfaces

The main objective of our system is to provide online student assessment service for not only instructors but also students themselves. The Web application consists of three principle interfaces, including:

1. Student performance within chapters, which displays the weekly knowledge states of all students who have once submitted quizzes.
2. Course overview in multi-granularity level, which shows the distribution of students having different mastery of knowledge within a chapter or a specific knowledge component.
3. Individual knowledge state, which visualizes one's knowledge state of different knowledge components and compares it with the average level among the whole class.

We display several screenshots of three interfaces and describe how they interact with users and help to improve MOOC learning.

Both the first and second interfaces can be useful for instructors to know how well students master the course content under different granules. Based on these results, instructors can take appropriate measures to improve student learning effect, such as remaking video lectures, supplementing additional readings, explaining for difficult knowledge components, revising confusing questions and providing guidance for individuals. The last interface displays the comparison of knowledge states between individuals and the average among the class, which helps students to understand their own weak points. Based on such comparison, students can review related learning materials and then understand the mistakes they made during the quizzes. Moreover, students may realize why they get stuck with the upcoming course content, since they have not mastered the prerequisite knowledge components yet.

5.3.1 Student performance

The first interface displays a table that lists the general knowledge states, i.e. probability of achieving the knowledge within different chapters, of students who submitted at least one assignment, as shown in Figure 11. This page consists of three component, detailed knowledge states of each student, a chapter list and a student list, as annotated in the figure. To have a more general look at the performance of students, the chapter list provide access to pages of course overview within a specific chapter. Besides, instructors can further understand how a student perform on different knowledge components by clicking on the specific student, which links to the page of individual knowledge states.

These information can be useful for instructors to discover students who did not perform well and then take appropriate interventions to improve their learning results. For example, we find students like 4136964 or 5087574 even fail to meet the requirement in the first chapter. So, we encourage the instructor to provide some basic content for them as additional readings. For students 5249026 and 3072009, they might be confused during the course and

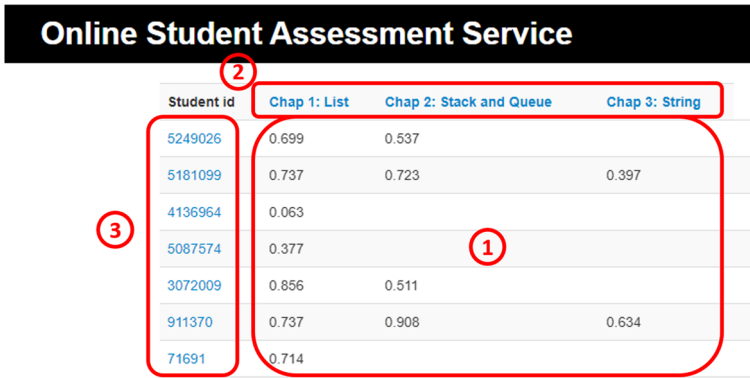


Figure 11 Detailed knowledge states of all students in the course

eventually drop out, so that the instructor may pay more attention to their posts and give some assists.

5.3.2 Course overview

In order to summarize the overall performance of students, we design an interface, as displayed in Figure 12, to plot the distribution of students with different knowledge states. We depict the performance of students from the perspectives of both chapters and knowledge components, so that instructors can not only glance over student performance within chapters, but also overview on a specific knowledge component, as shown in Figure 12a. Besides, instructors can also compare the student performance between different knowledge components, and then improve the courses by delivering more explanation for those knowledge components students are still confused with. Generally, such summarization can be a more intuitive way for instructors to understand how students have learned so far.

Figure 12a indicates three clusters of students and we simply call them the unlearned ones, well learned ones and partially learned ones, according to their knowledge states. Specifically, we find that the distributions of students with different mastery within chapters look like a multi-modal distribution, while those within knowledge components look like normal distribution, as Figure 12b and c shown. It may suggest that those partially learned students mastered some aspects well rather than partially learned every knowledge component.

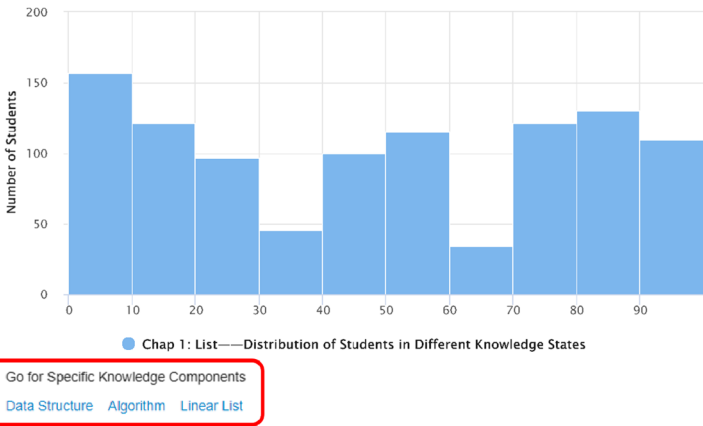
5.3.3 Individual knowledge states

Figure 13c shows the interfaces that display individual knowledge states, which leverage a radar chart to point out the mastery of each knowledge component involved in the quizzes. Besides, we also plot the average score of the class in order to compare individual knowledge states with other students, which makes them aware of their relative performance. According to the results of comparison, we list three typical kinds of students in Figure 13, which we called the unlearned, the partially learned and the well learned in Section 5.3.2.

The first type of students behaves like that described in Figure 13a, where they try to complete the quizzes but eventually fall behind others in terms of all the knowledge

Online Student Assessment Service

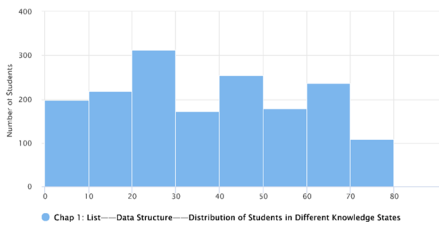
Chap 1: List—Distribution of Students



(a) Course overview in a specific chapter – List

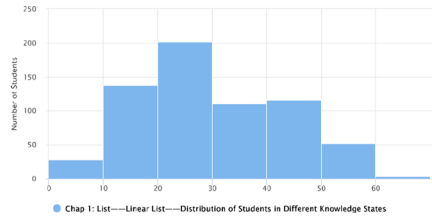
Online Student Assessment Service

Chap 1: List—Data Structure—Distribution of Students



Online Student Assessment Service

Chap 1: List—Linear List—Distribution of Students



(b) Knowledge component: Data Structure

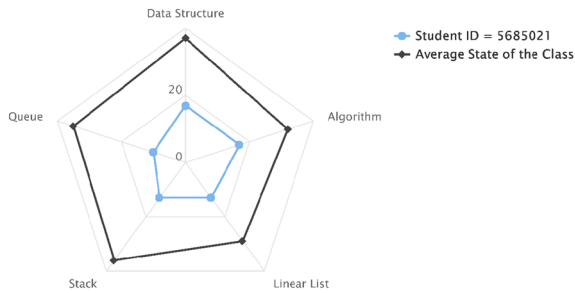
(c) Knowledge component: Linear List

Figure 12 The distribution of students having different mastery of knowledge

components they have learned. And that often leads to student dropout. One possible guidance for these students is to provide more fundamental learning materials which also contain explanation of these knowledge components. In contrast, some students have mastered all knowledge components well, like the student shown in Figure 13b. One potential guidance for these students can be some recommendations of further study, such as the extended content of the knowledge components, the applications of these knowledge in real datasets and more complicated exercises. The third type of students is a more complicated case where they perform better than the class average in terms of some knowledge components, while they perform worse in the other aspects. They may just want to learn a specific knowledge component, or put less emphasis on the basic concepts. And this interface can be useful for students to examine their mastery of a specific knowledge component.

Online Student Assessment Service

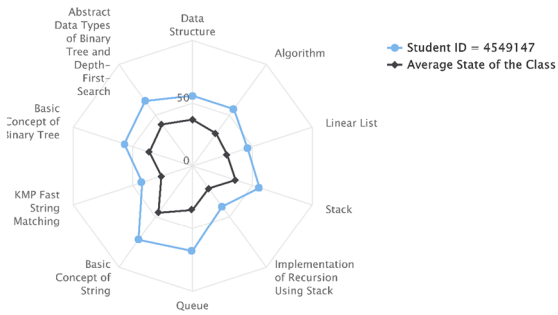
Individual Knowledge States



(a) Unlearned student

Online Student Assessment Service

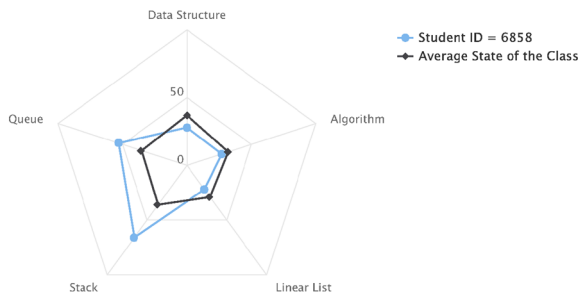
Individual Knowledge States



(b) Well learned student

Online Student Assessment Service

Individual Knowledge States



(c) Partially learned student

Figure 13 The comparison of students with different knowledge states

6 Conclusion and future work

In this paper, we study the problem of how to provide individual learning guidance for students with different engagement styles. We propose a novel framework that leverages multiple sources of data as inputs, then designs several algorithms to understand both the course and students, and finally visualize the result through a Web application. In terms of the models, we use a graph-based ranking algorithm to identify the key concepts in the course. We then propose two knowledge tracing models, Multi-Grained-BKT and Historical-BKT, to evaluate student knowledge states. We also design a dropout prediction system to understand the correlation between student behaviors and dropout. The experiments show the effectiveness of our algorithms and we discuss on the results both quantitatively and qualitatively. We think the content analysis may help students have an vision of key points before watching lectures, the assessment may suggest students their weak points after they try an assignment and the dropout prediction can give an early warning to those who want to complete the course. Moreover, we design a Web application to provide online student assessment service. We list the knowledge states of all students who have once submitted quizzes. We then display the course overview of student performance within a chapter or a specific knowledge component. We also plot both individual knowledge states and the average of the whole class for comparison.

However, our work is just an initial solution for personalized intervention in MOOCs. There are still some issues remained as scopes of the future work. For example, when identifying the key concepts from the course, we can take the similarity of word embeddings to construct the graph instead of word co-occurrence. Besides, it may be effective to incorporate the learning behavior data with knowledge tracing model, for MOOC students not only learn by doing exercises but also watching lectures. Except for the interpretability, recurrent neural networks have shown great promise to model sequential data, which can be used to predict student dropout in MOOCs.

Acknowledgements This paper is partially supported by the National Natural Science Foundation of China (NSFC Grant Nos.61472006, 61772039, and 91646202).

References

1. Agrawal, A., Venkatraman, J., Leonard, S., Paepcke, A.: Yuedu: addressing confusion in Mooc discussion forums by recommending instructional video clips. In: *Educational Data Mining 2015*, pp. 297–304 (2015)
2. Anderson, A., Huttenlocher, D., Kleinberg, J., Leskovec, J.: Engaging with massive online courses. In: *Proceedings of the 23rd International Conference on World Wide Web*, pp. 687–698. ACM (2014)
3. Basu, S., Wu, A., Hou, B., DeNero, J.: Problems before solutions: automated problem clarification at scale. In: *Proceedings of the Second ACM Conference on Learning@ Scale*, pp. 205–213. ACM (2015)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. *J. Mach. Learn. Res.* **3**(Jan), 993–1022 (2003)
5. Bougouin, A., Boudin, F., Daille, B.: Topicrank: graph-based topic ranking for keyphrase extraction. In: *International Joint Conference on Natural Language Processing (IJCNLP)*, pp. 543–551 (2013)
6. Breslow, L., Pritchard, D.E., DeBoer, J., Stump, G.S., Ho, A.D., Seaton, D.T.: Studying learning in the worldwide classroom: Research into edx’s first mooc. *Res. Pract. Assess.* **8**, 13–25 (2013)
7. Chaturvedi, S., Goldwasser, D., Daumé, H. III.: Predicting instructor’s intervention in Mooc forums. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 1501–1511. ACL (2014)
8. Chen, Y., Zhang, M.: Mooc student dropout: pattern and prevention. In: *Proceedings of the ACM Turing 50th Celebration Conference-China*, pp. 4:1–4:6. ACM (2017)

9. Conole, G.: Moocs as disruptive technologies: strategies for enhancing the learner experience and quality of moocs. *Revista de Educación a Distancia* (39), 1–17 (2015)
10. Corbett, A.T., Anderson, J.R.: Knowledge tracing: modelling the acquisition of procedural knowledge. *User Model. User-Adap. Inter.* **4**(4), 253–278 (1995)
11. Davis, D., Chen, G., Van der Zee, T., Hauff, C., Houben, G.J.: Retrieval practice and study planning in Moocs: exploring classroom-based self-regulated learning strategies at scale. In: *European Conference on Technology Enhanced Learning*, pp. 57–71. Springer, New York (2016)
12. Guo, P.J., Kim, J., Rubin, R.: How video production affects student engagement: an empirical study of Mooc videos. In: *Proceedings of the First ACM Conference on Learning@ Scale Conference*, pp. 41–50. ACM (2014)
13. Hmedna, B., El Mezouary, A., Baz, O., Mammass, D.: Identifying and tracking learning styles in moocs: a neural networks approach. *Int. J. Innov. Manag. Appl. Stud.* **19**(2), 267 (2017)
14. Ho, T.K.: Random decision forests. In: *Proceedings of the Third International Conference on Document Analysis and Recognition*, vol. 1, pp. 278–282. IEEE (1995)
15. Jiang, Z., Zhang, Y., Li, X.: Moocon: a framework for semi-supervised concept extraction from Mooc content. In: *International Conference on Database Systems for Advanced Applications*, pp. 303–315. Springer, New York (2017)
16. Jing, X., Tang, J.: Guess you like: course recommendation in Moocs. In: *Proceedings of the International Conference on Web Intelligence*, pp. 783–789. ACM (2017)
17. Jordan, K.: Initial trends in enrolment and completion of massive open online courses. *IRRODL* **15**(1), 133–160 (2014)
18. Khajah, M., Lindsey, R.V., Mozer, M.C.: How deep is knowledge tracing? In: *Educational Data Mining 2016*, pp. 94–101 (2016)
19. Khosravi, H., Cooper, K., Kitto, K.: Riple: Recommendation in peer-learning environments based on knowledge gaps and interests. *JEDM* **9**(1), 42–67 (2017)
20. Kim, T.d., Yang, M.y., Bae, J., Min, B.A., Lee, I., Kim, J.: Escape from infinite freedom: effects of constraining user freedom on the prevention of dropout in an online learning context. *Comput. Hum. Behav.* **66**, 217–231 (2017)
21. Kizilcec, R.F., Pérez-Sanagustín, M., Maldonado, J.J.: Recommending self-regulated learning strategies does not improve performance in a Mooc. In: *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, pp. 101–104. ACM (2016)
22. Kolowich, S.: Coursera takes a nuanced view of mooc dropout rates the chronicle of higher education (2013)
23. Kovacs, G.: Effects of in-video quizzes on Mooc lecture viewing. In: *Proceedings of the Third ACM Conference on Learning@ Scale*, pp. 31–40. ACM (2016)
24. Kulik, J.A., Kulik, C.L.C.: Timing of feedback and verbal learning. *Rev. Educ. Res.* **58**(1), 79–97 (1988)
25. Kulkarni, C.E., Bernstein, M.S., Klemmer, S.R.: Peerstudio: rapid peer feedback emphasizes revision and improves performance. In: *Proceedings of the Second ACM Conference on Learning@ Scale*, pp. 75–84. ACM (2015)
26. Li, W., Gao, M., Li, H., Xiong, Q., Wen, J., Wu, Z.: Dropout prediction in Moocs using behavior features and multi-view semi-supervised learning. In: *International Joint Conference on Neural Networks (IJCNN)*, pp. 3130–3137. IEEE (2016)
27. Liu, C.C., Chang, C.J., Tseng, J.M.: The effect of recommendation systems on internet-based learning for different learners: a data mining analysis. *Br. J. Educ. Technol.* **44**(5), 758–773 (2013)
28. Matsuda, N., Furukawa, T., Bier, N., Faloutsos, C.: Machine beats experts: automatic discovery of skill models for data-driven online course refinement. In: *Educational Data Mining 2015*, pp. 101–108 (2015)
29. Mihalcea, R., Tarau, P.: Textrank: bringing order into Texts. In: *Conference on Empirical Methods in Natural Language Processing*, pp. 404–411. ACL (2004)
30. Onah, D.F., Sinclair, J., Boyatt, R.: Dropout rates of massive open online courses: behavioural patterns. In: *International Conference on Education & New Learning Technologies*, pp. 5825–5834 (2014)
31. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the Web. *Tech. Rep., Stanford InfoLab* (1999)
32. Pan, L., Li, C., Li, J., Tang, J.: Prerequisite relation learning for concepts in Moocs. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 1447–1456. ACL, Vancouver (2017)
33. Parameswaran, A., Garcia-Molina, H., Rajaraman, A.: Towards the Web of concepts: Extracting concepts from large datasets. In: *Proceedings of the VLDB Endowment*, vol. 3, pp. 566–577 (2010)
34. Pardos, Z.A., Heffernan, N.T.: Modeling individualization in a Bayesian networks implementation of knowledge tracing. In: *International Conference on User Modeling, Adaptation, and Personalization*, pp. 255–266. Springer, New York (2010)

35. Pardos, Z., Heffernan, N.: Kt-idem: introducing item difficulty to the knowledge tracing model. In: *User Modeling, Adaption and Personalization*, pp. 243–254 (2011)
36. Pardos, Z., Bergner, Y., Seaton, D., Pritchard, D.: Adapting Bayesian knowledge tracing to a massive open online course in Edx. In: *Educational Data Mining 2013*, pp. 939–951 (2013)
37. Pardos, Z.A., Tang, S., Davis, D., Le, C.V.: Enabling real-time adaptivity in Moocs with a personalized next-step recommendation framework. In: *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*, pp. 23–32. ACM (2017)
38. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Deep knowledge tracing. In: *Advances in Neural Information Processing Systems*, pp. 505–513 (2015)
39. Qiu, J., Tang, J., Liu, T.X., Gong, J., Zhang, C., Zhang, Q., Xue, Y.: Modeling and predicting learning behavior in Moocs. In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pp. 93–102. ACM (2016)
40. Ramesh, A., Goldwasser, D., Huang, B., Daumé, H. III., Getoor, L.: Modeling learner engagement in Moocs using probabilistic soft logic. In: *NIPS Workshop on Data Driven Education*, vol. 21, pp. 62 (2013)
41. Reye, J.: Student modelling based on belief networks. *Int. J. Artif. Intell. Educ.* **14**(1), 63–96 (2004)
42. Ritter, S., Anderson, J.R., Koedinger, K.R., Corbett, A.: Cognitive tutor: applied research in mathematics education. *Psychon. Bull. Rev.* **14**(2), 249–255 (2007)
43. Sonwalkar, N.: The first adaptive Mooc: a case study on pedagogy framework and scalable cloud architecture - Part I. In: *MOOCs Forum*, vol. 1, pp. 22–29 (2013)
44. Sunar, A., Abdullah, N., White, S., Davis, H.: Personalisation of Moocs: the state of the art. In: *CSEDU 2015 - 7Th International Conference on Computer Supported Education, Proceedings*, vol. 1, pp. 88–97 (2015)
45. Tomkin, J.H., Charlevoix, D.: Do Professors matter?: using an A/B test to evaluate the impact of instructor involvement on Mooc student outcomes. In: *Proceedings of the First ACM Conference on Learning@ Scale Conference*, pp. 71–78. ACM (2014)
46. Wang, Z., Zhu, J., Li, X., Hu, Z., Zhang, M.: Structured knowledge tracing models for student assessment on Coursera. In: *Proceedings of the Third ACM Conference on Learning@ Scale*, pp. 209–212. ACM (2016)
47. Xing, W., Chen, X., Stein, J., Marcinkowski, M.: Temporal predication of dropouts in moocs: Reaching the low hanging fruit through stacking generalization. *Comput. Hum. Behav.* **58**, 119–129 (2016)
48. Yang, D., Sinha, T., Adamson, D., Rosé, C.P.: Turn on, tune in, drop out: anticipating student dropouts in massive open online courses. In: *Proceedings of the 2013 NIPS Data-Driven Education Workshop*, vol. 11, pp. 14 (2013)
49. Yang, D., Piergallini, M., Howley, I., Rose, C.: Forum thread recommendation for massive open online courses. In: *Educational Data Mining 2014*, pp. 257–260 (2014)
50. Yang, D., Kraut, R., Rosé, C.P.: Exploring the effect of student confusion in massive open online courses. *J. Educ. Data Min.* **8**(1), 52–83 (2016)
51. Yudelson, M.V., Koedinger, K.R., Gordon, G.J.: Individualized Bayesian knowledge tracing models. In: *International Conference on Artificial Intelligence in Education*, pp. 171–180. Springer, New York (2013)
52. Zhu, J., Li, X., Wang, Z., Zhang, M.: An effective framework for automatically generating and ranking topics in Mooc videos. In: *Educational Data Mining 2017*, pp. 150–155 (2017)