Automatically Learning Topics and Difficulty Levels of Problems in Online Judge Systems

WAYNE XIN ZHAO, Renmin University of China
WENHUI ZHANG, Renmin University of China
YULAN HE, Aston University
XING XIE, Microsoft Research
JI-RONG WEN, Renmin University of China

Online Judge (OJ) systems have been widely used in many areas, including programming, mathematical problems solving and job interviews. Unlike other online learning systems such as MOOC, most OJ systems are designed for self-directed learning without the intervention of teachers. Also, in most OJ systems, problems are simply listed in volumes and there is no clear organization of them by topics or difficulty levels. As such, problems in the same volume are mixed in terms of topics or difficulty levels. By analyzing large-scale users' learning traces, we observe that there are two major learning modes (or patterns). Users either practice problems in a sequential manner from the same volume regardless of their topics or they attempt problems about the same topic, which may spread across multiple volumes. Our observation is consistent with the findings in classic educational psychology. Based on our observation, we propose a novel two-mode Markov topic model to automatically detect the topics of online problems by jointly characterizing the two learning modes. For further predicting the difficulty level of online problems, we propose a competition-based expertise model using the learned topic information. Extensive experiments on three large OJ datasets have demonstrated the effectiveness of our approach in three different tasks, including skill topic extraction, expertise competition prediction and problem recommendation.

CCS Concepts: • **Information systems** → *Collaborative and social computing systems and tools*; *Collaborative filtering*; *Social recommendation*; • **Computing methodologies** → *Topic modeling*; • **Applied computing** → *Education*;

Additional Key Words and Phrases: Topic models, Expertise learning, Online judge systems

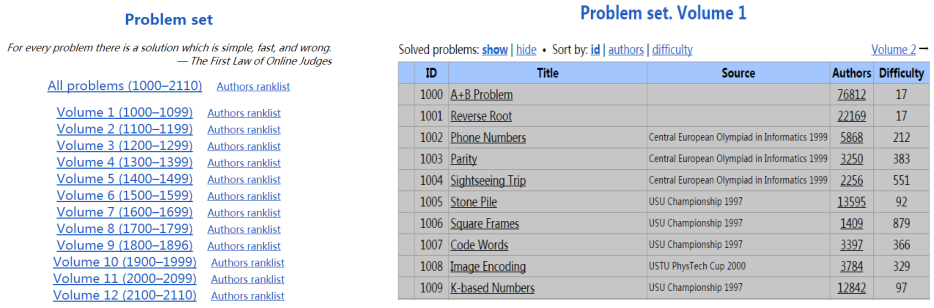
ACM Reference format:
WAYNE XIN ZHAO, WENHUI ZHANG, YULAN HE, XING XIE, and JI-RONG WEN. 2018. Automatically Learning Topics and Difficulty Levels of Problems in Online Judge Systems. *ACM Transactions on Information Systems* 36, 3, Article 27 (April 2018), 34 pages.
DOI: 0000001.0000001

The work was partially supported by National Natural Science Foundation of China under Grant No. 61502502, the National Key Basic Research Program (973 Program) of China under Grant No. 2014CB340403 and Beijing Natural Science Foundation under Grant No. 4162032. The work was done when Xin Zhao visited Microsoft Research.
Authors' addresses: W. X. Zhao, W. Zhang (co-first author), and J.-R. Wen (corresponding author), School of Information & Beijing Key Laboratory of Big Data Management and Analysis Methods, Renmin University of China, Beijing 100872, China; emails: {batmanfly, jirong.wen}@gmail.com; Y. He, School of Engineering and Applied Science, Aston University, Birmingham, B4 7ET, United Kingdom; email: y.he9@aston.ac.uk; X. Xie, Microsoft Research, Beijing 100190, China; email: xingx@microsoft.com.
 1046-8188/2018/4-ART27 $15.00
DOI: 0000001.0000001

1 INTRODUCTION

The rapid advancement of Internet technologies largely promotes the development of online education services, which makes educational resources more easily accessible than ever before. Among these services, Massive Open Online Course (MOOC) platforms (e.g., Coursera) are maintained by experts and teachers, which support the interactions among students, professors, and teaching assistants. On the contrary, Online Judge (OJ) systems are designed for self-directed learning without interventions from or interactions with teachers. OJ systems usually maintain a large pool of problems or questions (a.k.a. *problem bank*) related to some specific domain, and provide real-time automatic assessment of the solutions submitted by users.

OJ systems can serve as an open and shared test platform with increasingly new resources for self-regulated learning [50]. Example OJ systems include mathematical problem solving sites like ACT Math[1], job interview practice sites like LeetCode[2], and driver license sites like JKYDT[3]. On OJ systems, learning resources (i.e., problems) are often organized by *volumes*; a fixed number of problems are selected and piled into a volume. Each problem from a volume is assigned with a unique ID. This essentially forms a two-layer index structure for problems: volume index and problem ID. We present the snapshots of Timus Online Judge[4] in Fig. 1, which is the largest Russian archive of programming problems, to illustrate the organization of problems by volumes. Up to date, Timus contains 1,110 problems uniformly distributed in 12 volumes. After logging into the system, a user can select any problem for practice from one of the 12 volumes. Using volume organization, there are low maintenance costs for developing and updating an OJ system for repaid resource sharing and exercise assessment.



(a) Volume index. (b) Prolem index for Volume 1.

Fig. 1. Snapshots of the Timus Online Judge.

Volume organization is hard for users to locate required resources quickly. For example, a novice programmer only wants to practice easy programming problems, but it would take her considerable amount of time to identify suitable problems from the large problem pool at her ability level. Similarly, a user who only wants to focus on *dynamic programming* would also find it hard to locate appropriate problems as they may spread across multiple volumes. As have been previously

[1] http://http://www.act.org/content/act/en/products-and-services/the-act/test-preparation/english-practice-test-problems.html
[2] https://leetcode.com
[3] http://www.jkydt.com/
[4] http://acm.timus.ru/

observed, two very important factors should be considered in curricula design [6, 54, 57], namely, *topics* and *difficulty levels*. *Topics* refer to the categorization or grouping of problems in terms of their related knowledge components, concepts or skills; *difficulty levels* refer to the degrees of difficulty of problems. Traditionally, topics and difficulty levels are manually annotated for problems by domain experts or experienced teachers. It requires heavy manual efforts. Nevertheless, as have been shown in past studies [11, 16], users' learning traces often contain important evidence for inferring the information of topics and difficulty levels of problems. Hence, in this paper, we focus on mining users' learning traces for automatic detection of topics and difficulty levels of problems in OJ systems.

By analyzing users' learning traces, we find that there are two major learning modes. Users either choose problems in a sequential manner from the same volume regardless of their topics (*volume-oriented learning*) or focus on one specific topic and choose problems from multiple volumes but all about the same topic (*topic-oriented learning*). To characterize the two learning modes, we propose a novel two-mode Markov topic model, which can jointly characterize both learning patterns. We assume there are two sets of topics: *volume topics* characterize the generative probability of drawing a problem from a volume, while *skill topics* characterize the generative probability of drawing a problem from a skill topic. A user is profiled with two topic distributions over either volume topics or skill topics, and she has the preference to switch between the two modes. The sequential relatedness and mode selection preference are characterized in a joint model.

Apart from detecting topics of problems in OJ systems, we also study how to infer the problem difficulty and user expertise using users' learning traces. Different from traditional expertise models [13, 33], which usually characterize the difficulty or expertise level using a single scalar value, we have found that questions from different topics may have varying degrees of difficulties. Furthermore, a user is also likely to have different expertise levels on different topics. These findings indicate that difficulty or expertise levels should be defined over topics. Hence, we further propose a topic-aware competition-based expertise model. We assume that a problem is associated with a topic-specific difficulty score, while a user is also associated with a topic-specific expertise score. If considering a problem as a virtual user, then both difficulty or expertise levels can be characterized by a vector of topic-specific expertise scores in the same representation space. We model the problem solving process as a competition between a user and a problem. The topic information of a problem is obtained using the previously proposed two-mode Markov topic model, and incorporated as the context of the competition for learning the difficulty (or expertise) levels.

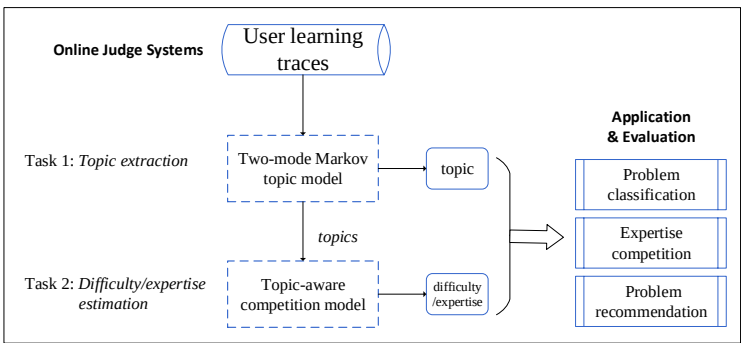


Fig. 2. Overview of the proposed approach. The solutions to task 1 and task 2 are presented in Section 4.1 and 4.2 respectively. We further present the applications of the proposed approach in Section 4.3 and experimental results in Section 5.

As shown in Fig. 2, we focus on two tasks, *topic detection* which automatically identifies the topics of a given problem and *difficulty/expertise level estimation* which quantizes the difficulty level of a problem. The derived topic and difficulty/expertise information can be subsequently used in downstream OJ-related applications, e.g., problem recommendation or expert finding. Our proposed approaches have been evaluated on the programming OJ systems, but they are also applicable to any other OJ systems. Our contributions are summarized as follows:

- We present the first study to automatically learn topics and difficulty levels for problems on OJ systems where problems are simply organized by volumes.
- We quantitatively analyze the learning behaviors of users on OJ systems. We have identified two learning patterns, i.e., volume-oriented and topic-oriented learning. The observation is further explained and supported by the theories from educational psychology. Based on the above behavioral characteristics, we propose a two-mode hidden Markov topic model for learning topics of problems.
- We quantitatively analyze the distribution of difficulty levels over different topics. We have found that questions from different topics may have varying degrees of difficulty levels. Based on this finding, we develop a topic-specific expertise model for estimating problem difficulty levels and user expertise scores.
- Extensive experiments on three large online judge datasets have demonstrated the effectiveness of our approach in three different tasks, including skill topic extraction, expertise competition prediction and problem recommendation.

2 PROBLEM DEFINITION

Let u and q denote a user and a problem respectively. Assume that there are a set of users $\mathcal{U}$ and a set of problems $\mathcal{Q}$. Usually, the problems are organized in a volume set $\mathcal{V}$. Each problem q belongs to a unique volume $v_q \in \mathcal{V}$. In an OJ system, a user u can submit her solution to a problem q. We call such an action an *attempt*. Each attempt happens at a timestamp s and ends with a status label l. The timestamps are accurate to the second in the UNIX timing system, and the status label usually only has two possible values: Pass or Failed. Given a user u, we sort her N_u attempt records in an ascending order according to the timestamps, and obtain an ordered *User Attempt Sequence* (UAS): $\langle q_1, s_1, l_1\rangle \rightarrow \langle q_2, s_2, l_2\rangle \rightarrow ... \rightarrow \langle q_i, s_i, l_i\rangle \rightarrow ... \rightarrow \langle q_{N_u}, s_{N_u}, l_{N_u}\rangle$, where we have $s_1 < s_2 < ... < s_i < ... < s_{N_u}$. With user attempt sequences, we can trace the attempts of a user over time.

The attempt label reflects the individual competition result between a user and a problem. We further introduce a competition between two users on a problem. An *Expertise Competition Triplet (ECT)* $\langle u, u', q\rangle$ consists of two users u, u' and a problem q. Given a pair of users (u, u'), we can form an ECT $\langle u, u', q\rangle$ if there exists some intermediate problem q that has been successfully solved by u but not by u'. That is to say, u beats u' on the problem q. To make the notations consistent, we create a virtual user u_q for each problem q. Our final user set $\mathcal{U}$ is extended to the union set of the user set and problem set, i.e., $\mathcal{U}' = \mathcal{U} \cup \mathcal{Q}$. In this way, $\langle u, u_q, q\rangle$ can be used to represent the competition between a user u and a problem q.

Using the above formulation, user attempt records are characterized in two kinds of data representations: the user-specific attempt sequences which model user attempts in a sequential manner, and the problem-specific competition triplets which model the comparison between two users (including virtual users corresponding to problems) in terms of their expertise levels. We could potentially also consider using problem descriptions to extract topic information. However, problem descriptions often contain figures or attachments which makes it hard to extract relevant information. Hence they are ignored in our work here.

With the above definitions, we are ready to define our two tasks, namely *topic extraction* and *difficulty/expertise level estimation*.

Definition 2.1. ***Topic Extraction***. Topics here refer to the related components, skills and facts for structuring and organizing problems in a coherent way. The goal of topic extraction is to discover a set of skill topics $\mathcal{T}$ in OJ systems, and each topic z is a multinomial distribution over the problems $\{\phi_{z,q}\}_{q \in Q}$.

For example, a sectional organization of C LANGUAGE PROGRAMMING can be listed as follows: *introduction*, *variables*, *loops and controls*, *arrays*, *struct* and *pointers* and *IO*. These sections can be considered as topics in a coarse granularity. It is worth noting that topic granularity is not pre-defined but automatically learned from data.

Definition 2.2. ***Difficulty/Expertise Level Estimation***. The goal of difficulty/expertise level estimation is to infer the difficulty/expertise level of a problem or a user in OJ systems. Specially, given a user u (or problem q), it aims to estimate the expertise level of u on each specific skill topic z, denoted by $s_{u,z}$, which is a real scalar value in some predefined range. Similarly, we can use $s_{q,z}$ to denote the difficulty level of problem q on topic z.

As discussed in [6, 57], topics and difficulty levels are very important to be used to organize educational resources. Hence, our major focus of this work is to study how to learn them automatically based on users' learning traces. To solve the above two tasks, we propose a topic modeling approach to learn topics and difficulty levels (Section 4.1 and 4.2). Furthermore, we can utilize the identified topics and difficulty levels information to improve related applications in OJ systems such as *problem recommendation* (Section 4.3).

3 DATA COLLECTION AND ANALYSIS

We select three popular programming OJs for analysis and evaluation:

- *Timus*: Timus Online Judge is the largest Russian archive of programming problems with automatic judging system. Questions are mostly collected from contests held at the Ural Federal University, Ural Championships, Ural ACM ICPC Subregional Contests, and Petrozavodsk Training Camps.
- *POJ*: Peking University Online Judge[5] is the most famous Chinese archive of programming problems. POJ has attracted the best programmers in China and even worldwide.
- *HDU*: HDU[6] is hosted by Hangzhou Dianzi University from China. It consists of a large number of problems for beginners, many of which are presented in Chinese.

On all the three platforms, problem resources are organized by volumes. OJ systems provide automatic assessment to users' submitted solutions in a black-box test way. A problem consists of multiple test cases with the same input and output format. Only after a user's submitted programming code passes all the test cases, the system would return the *"Accepted"* status, indicating the problem has been solved successfully. Programming OJ systems present in real-time a ranked list of users based on their number of accepted (a.k.a. *solved*) problems. The entire attempt record of a user is visible to all the other users. Such an open and competitive mechanism is one of the key successful factors for programming OJ systems [50].

[5]http://poj.org/
[6]http://http://acm.hdu.edu.cn/

Table 1. Statistics of our datasets.

Dataset	# Users	# Questions	# Attempts	# Triplets	#Ave. Attempted	#Ave. Solved
Timus	9,973	1,099	1,419,786	2,089,651	142	65
POJ	20,100	3,054	4,755,216	8,304,625	237	112
HDU	19,410	1,999	3,477,657	6,145,544	179	95

3.1 Construction of the Datasets

We crawl the data from the three OJ systems. At the time of crawling, the total numbers of registered users in Timus, POJ and HDU are 111,182, 715,002 and 460,322 respectively. Most of the tail users have made very few submissions, so we only keep the top users ranked by the number of accepted problems. To construct the user attempt sequence, we remove repeated attempts from a user on the same problem in a very short time period (i.e., 1 hour), and only keep the latest attempt with the Pass label. To construct the expertise competition triplets, we only use the final attempt result, i.e., once u has successfully solved q regardless of how many times she has failed previously, u is considered to be able to beat q. The statistics of our datasets is presented in Table 1.

Timus provides category (or topic) labels for some problems and difficulty scores for all problems. Out of 1,110 problems, 538 are tagged with one or multiple category labels from the category set { *Dynamic programming, Palindromes, Geometry, Tricky problem, Hardest problem, Data structures, Number theory, Graph theory problem for beginners, Game, Unusual problem, String algorithms* }. For POJ and HDU datasets, there are no such labeled data. Nevertheless, we notice that some top performing users categorize the problems they have solved by topics in their blogs. We therefore use those category labels as the topic labels of problems. For problems with multiple category labels assigned by more than one user, we manually merge those labels. Finally, 458 out of 3,054 problems on POJ and 482 out of 1,999 problems on HDU have been annotated with a topic label (which was proposed by at least two users) using a similar category set from Timus. We do not annotate the difficulty levels for the problems on POJ and HDU, since they are not used explicitly as ground truth in our evaluations.

3.2 Basic Analysis and Observations

Given the datasets, we conduct some initial analysis on the Timus dataset. Our findings on the other two datasets are similar and are hence omitted here. First, we plot the distributions of the number of users with respective to their numbers of attempted or accepted problems in Fig. 3. We can observe that both distributions show a clear power-law like shape, which indicates that only a small fraction of users have attempted or solved many problems.

We then conduct further analysis related to topics and difficulty levels and make the following observations.

Observation 1: *A user is likely to make consecutive attempts on problems from the same volume.* Since problems are organized by volumes, we first examine whether the user behaviors are affected by such an organization. For each user, we sample thirty short windows consisting of five attempts (we remove consecutive repetitive submissions). Then we make pairwise comparisons between two problems from a window and check whether they belong to the same volume. We have found that about 40.2% problem pairs are indeed from the same volume. In a random setting, two problems from the same volume can be approximately estimated by $\frac{V}{V \times V}$, where $V = 11$ is the number of volumes on the Timus dataset. Hence, the reference ratio is roughly 9.0% by chance. We can see that the derived ratio (40.2%) is significantly higher than the reference ratio (9.0%), which indicates that the learning behavior of a user is indeed affected by volume organization.

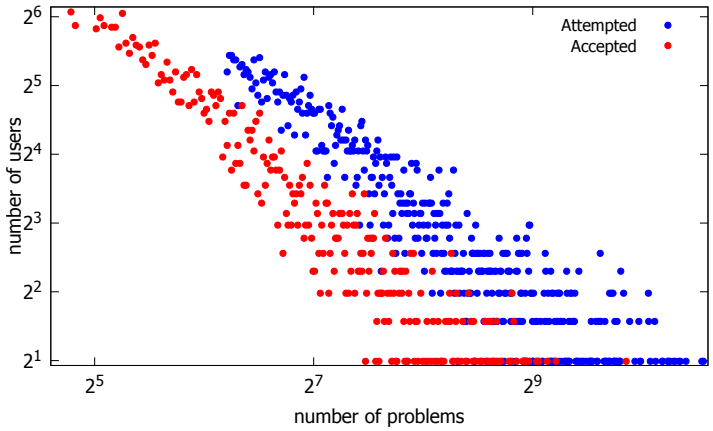


Fig. 3. Distributions of users *w.r.t.* the number of accepted or attempted problems on the Timus dataset in log-log scale. We have excluded users with few attempts for ease of data crawling.

Observation 2: *A user is likely to make consecutive attempts on problems under the same topic.* Similar to the above analysis, we continue to study the effect of the topic information on users' learning behavior. For each user, we sample thirty short windows consisting of five attempts (we remove consecutive repetitive submissions). Then we make pairwise comparisons between two labeled problems (problems without labels are ignored here) from a window and check whether they belong to the same topic category. We have found that about 41.4% problem pairs are indeed labeled with the same topic. The ratio is also significantly larger than the reference ratio 8.3%, which is estimated by $\frac{T}{T \times T}$ and $T(= 12)$ is the total number of topic categories on Timus. Unlike the analysis in *Obs. 1*, the estimated ratio here is only an approximation, since only half of the problems were annotated with the topic labels. Although the problems are organized by volumes without any topic information, the users still show a strong learning pattern by working on the same topic consecutively.

Observation 3: *Questions from different topics may have varying degrees of difficulty levels.* Here we analyze the characteristics of the distribution of difficulty levels for problems under different topics. We plot the distributions over 12 topic categories in Figure 4. It can be observed that the distribution of difficulty scores highly depends on topics.

We can explain *Obs.* 1 and 2 using classic theories in educational psychology. In classic theories on the arrangement of practice sets, problems are usually arranged in two ways, namely, *blocked* where all problems are drawn from the same topic, and *mixed* where a mixture of problems are drawn from multiple topics [51]. The volume organization can be considered a specific case of the *mixed* practice, where each volume contains a mixed set of problems in different topics. The behavior of repeatedly practicing problems on the same topic is called *overlearning* [52]. Our finding indicates that even with a mixed practice setting, a user could still show the overlearning behavior as typically found in the blocked setting. Such a phenomenon can be explained by the classic metacognition learning theory [56], which describes the processes involving when learners plan, monitor, evaluate and make changes to their own learning behaviors. The above findings show that users typically adopt a mixture of the two learning patterns (we call it *two-mode learning behaviors*). An example of two-mode learning behaviors is illustrated in Figure. 5. In such a trace window, the

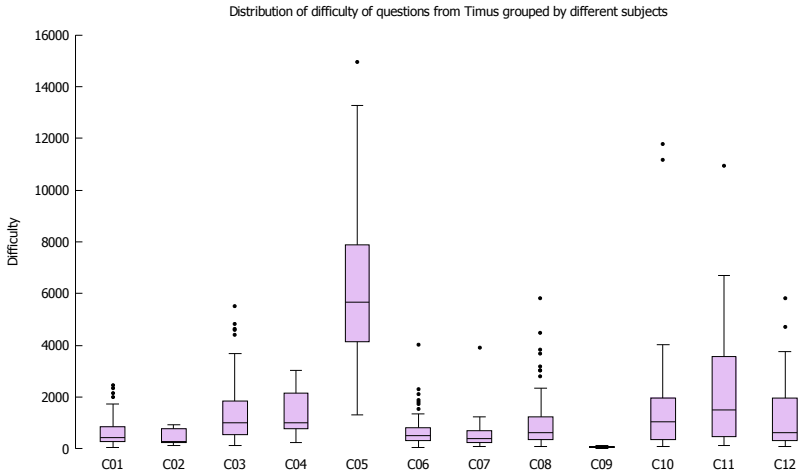


Fig. 4. The distribution of difficulty scores by topics on the Timus dataset. For ease of visualization, we use C*i* to index the categories. C01~C12 refer to Dynamic programming, Palindromes, Geometry, Tricky problem, Hardest problem, Data structures, Number theory, Graph theory problem for beginners, Game, Unusual problem, String algorithms respectively. A larger score indicates a higher difficulty level.

user with nickname of *momohuang* has attempted 12 problems. It can be seen that the first seven attempts are topic-oriented, while the last five attempts are volume-oriented. *Obs.* 1 and 2 also motivate us to develop a two-mode Markov topic model to characterize users' learning behaviors for topic extraction (Section 4.1).

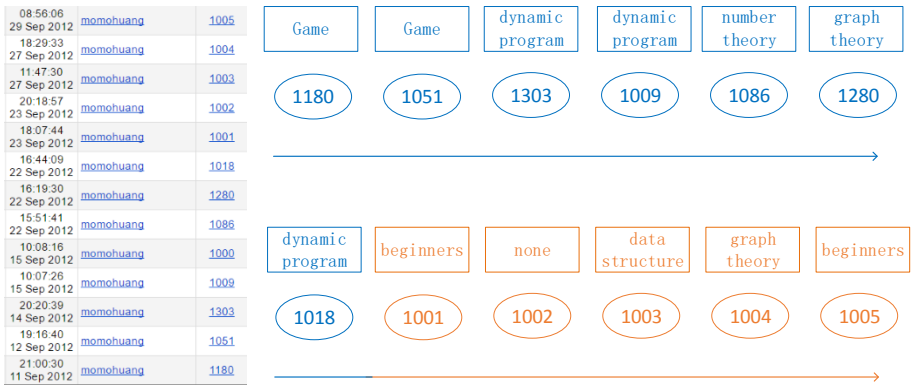


Fig. 5. Illustrations of the two-mode learning behaviors from a sample user on Timus. Such an attempt sequence is called *users' learning trace*. It is evident that the last five consecutive attempts are from Volume I. Although the seventh attempt is also from Volume 1, it is likely to be topic-driven since there have already been two attempts on dynamic programming.

As for *Obs.* 3, we have observed that the difficulty levels of problems from different topics are different and varying. We could utilize the topic information of problems for better characterizing difficulty levels of problems. *Obs.* 3 thus motivates us to develop an approach to model problem difficulty and user expertise levels (Section 4.2).

4 A TOPIC MODELING APPROACH FOR LEARNING TOPICS AND DIFFICULTY LEVELS

In Section 3.2, we have made some interesting observations about topics and difficulty levels. In this section, we study how to develop our solution for topic extraction and difficulty level estimation based on these findings. A key point is that the entire learning process should be unsupervised and rely on no or little labeled data, since it is the actual case for most of OJ systems. Hence, we adopt the topic modeling approach, which is able to generate meaningful soft clusters for "words", i.e., problems in our setting. Our approach can effectively extract skill topics, and further learn the topic-specific difficulty/expertise levels. In what follows, we start by studying how to learn topics in Section 4.1 and subsequently followed by the estimation of difficulty levels of problems in Section 4.2. In Section 4.3, we introduce some downstream applications that can be improved by our models. For clarity, we first present the notations used throughout the paper in Table 2.

4.1 Topic Extraction using a Two-Mode Hidden Markov Topic Model

Based on our definition, a user attempt sequence presents the learning trace consisting of her attempts on problems in a given time span, which is essentially a behavior sequence of user actions. We would like to capture the relatedness reflected in user behavior sequences, and infer topic information from there. Specifically, we adopt *topics* from topic modeling [5] to model the grouping or clustering of problems. As shown in *Obs.* 1 and 2, there are two different learning patterns to consider, either volume- or topic-oriented modes. To characterize both learning modes, we incorporate two kinds of topics in our model, namely volume topics and skill topics. Volume topics correspond to the volumes set up by OJ systems, while skill topics correspond to topics related to different skills. Since a problem attempt is likely to be influenced by both learning modes, we have to jointly characterize them in order to more accurately model the generative process for problems. We start to discuss how to model each type of relatedness separately, and then combine these two parts in a joint model.

4.1.1 Volume-Oriented Sequential Relatedness. As we can see in *Obs.* 1, a user is likely to make consecutive attempts on problems from the same volume. Such a behavioral pattern is similar to that of the click behavior [7] in the field of Information Retrieval. Since the problems are present to a user by volumes, once a user has attempted or solved a problem from a specific volume, the remaining problems from the same volume are more likely to attract the user's attention than those from other volumes. To capture this kind of behavioral relatedness, we introduce a set of *volume topics*. Formally, let ϕ_v^V denote the topic model corresponding to the volume v, where the superscript of V stands for "volume". The probability of $\phi_{v,q}^V$ denotes the conditional probability of a problem q from a volume topic v. Different from standard topic models, which have non-zero Bayesian priors over all the problems, we make a constraint on volume topics: only the problems from the volume v can be assigned with a non-zero probability by ϕ_v^V. Furthermore, each user u has a preference distribution over the volume topics, modeled by a multinomial distribution θ_u^V. In order to generate a problem from a volume topic, we first draw a volume topic label, and then generate the problem under the corresponding volume topic. The key point in capturing the sequential relatedness lies in modeling the dependency of volume labels from consecutive attempts. Following [24], we make the one-order Markov assumption that the topic label of the

Table 2. Notations and explanations.

Notations	Explanations
u	a user
N_u	number of attempts by user u
q	a problem
$\boldsymbol{q}_u$	the attempt sequence of user u
n	an index variable to the attempts
v	a volume or volume topic
z	a skill topic
c	binary continuity indicator
y	binary mode indicator
K^S	number of skill topics
K^V	number of volumes
ϕ_k^S	the k-th skill topic
ϕ_k^V	the k-th volume topic
θ_u^S	the preference distribution of u over the skill topics
θ_u^V	the preference distribution of u over the volume topics
ρ_u^S	the Bernoulli distribution over the continuation of skill topics
ρ_u^V	the Bernoulli distribution over the continuation of volume topics
$\boldsymbol{x}_{u,n}$	the features related to the n-th attempt from user u
$\boldsymbol{\epsilon}$	the weight vector of the logistic classifier
$e_u^{(q)}$	the expertise score of user u with the context of q
$\boldsymbol{x}_q$	the topical probabilities related to query q
b_u	the base expertise score of user u
$\boldsymbol{s}_u$	topic-specific expertise vector of user u
$\boldsymbol{w}$	the importance vector of topics
μ_u, σ_u	Gaussian parameters for user u
α, β, ψ	prior parameters of the topic models

current attempt depends on the topic label of the previous attempt. Formally, we assume that a user is associated with a Bernoulli distribution ρ_u^V. At the n-th attempt, a user first draws a binary indicator variable $c_{u,n} \sim Bern(\rho_u^V)$. If $c_{u,n} = 0$, we keep and reuse the volume topic label of the previous problem, i.e., $v_{q_{u,n}} = v_{q_{u,n-1}}$; otherwise, we draw a new topic label using the user-specific preference distribution θ_u^V.

4.1.2 Topic-Oriented Sequential Relatedness. We continue to model the topic-oriented sequential relatedness in *Obs.* 2. Formally, let $\mathcal{T}$ denote a set of skill topics, each of which is a multinomial distribution over the problems, denoted by ϕ_z^S, and the probability $\phi_{z,q}^S$, i.e., $Pr(q|z)$, indicates the conditional probability that a problem q is generated from topic z. We use the superscript of S to discriminate skill topics from volume topics. Furthermore, each user u has a preference distribution over a set of skill topics, modeled by a multinomial distribution θ_u^S, in which $\theta_{u,z}^S$ indicates the preference degree of user u on topic z. To generate a problem, we first sample a skill topic label, and then generate the problem using the corresponding skill topic model. We further make a similar Markov assumption as before: the current skill topic label depends on the skill topic label of the previous problem. We adopt the one-order Markov chain structure to characterize the topic relatedness. Formally, we assume that a user is associated with a Bernoulli distribution ρ_u^S. At the

Table 3. The different generation routes with the two-mode topic models. "follow" and "re-draw" refer to whether the previous topic assignment is reserved or newly sampled.

$y_{u,n}$	$y_{u,n-1}$	$c_{u,n}$	$z_{u,n}$	$v_{u,n}$	remark
1	1	1	–	$v \sim Multi(\theta_u^V)$	volume-mode, re-draw
1	1	0	–	$v_{u,n-1}$	volume-mode, follow
1	0	1	–	$v \sim Multi(\theta_u^V)$	volume-mode, re-draw
0	1	1	$z \sim Multi(\theta_u^S)$	–	topic-mode, re-draw
0	0	1	$z \sim Multi(\theta_u^S)$	–	topic-mode, re-draw
0	0	0	$z_{u,n-1}$	–	topic-mode, follow

n-th attempt, a user first draws a binary indicator variable $c_{u,n} \sim Bern(\rho_u^S)$. If $c_{u,n} = 0$, we keep and reuse the skill topic label of the previous problem, i.e., $z_{q_{u,n}} = z_{q_{u,n-1}}$; otherwise, we draw a new topic label using the user-specific preference distribution θ_u^S.

4.1.3 The Two-Mode Markov Topic Model. After describing the two kinds of learning patterns separately, we are ready to integrate them in a unified model. Our core idea is that a user has two optional learning modes to select, either volume-oriented or topic-oriented, indicating whether a user is likely to work on problems from the same volume or under the same skill topic. We use another hidden variable y to control the mode switch. The mode indicator variable $y_{u,n}$ is drawn from a user-specific Bernoulli distribution, denoted by η_u. The parameter η_u characterizes the tendency of user u to select over the two modes. With the incorporation of y, we can code the different generation routes in Table 3. Note that we use the same binary indicator variable c to model the Markovian relatedness for both kinds of topics. As we can see in Table 3, we can control and select the corresponding mode using a certain combination of y and c. Once these two variables are set, we can easily generate an attempted problem by a user. When determining the continuity variable c, we apply a hard constraint: once y has changed, c has to be re-drawn. Note that entries for the cases where $y_{u,n} \neq y_{u,n-1}$ and $c_{u,n} = 0$ are invalid, since the topic variable has to be re-drawn once the mode indicator has changed.

Table 4. The list of auxiliary features used in the logistic classifier for mode selection. $\mathbb{I}[\cdot]$ is an indicator function which returns 1 *iff* when the condition is true.

Mode feature	Formulation
local volume continuity	$\mathbb{I}[\tilde{y}_{u,n-1} = 1 \text{ and } \tilde{y}_{u,n+1} = 1]$
local skill continuity	$\mathbb{I}[\tilde{y}_{u,n-1} = 0 \text{ and } \tilde{y}_{u,n+1} = 0]$
forward volume continuity	$\arg\max_{k, k \leq 5, k \leq n}[\tilde{y}_{u,n-k} = \ldots = \tilde{y}_{u,n-1} = 1]$
backward volume continuity	$\arg\max_{k, k \leq 5, k+n \leq N_u}[\tilde{y}_{u,n+1} = \ldots = \tilde{y}_{u,n+k} = 1]$
forward skill continuity	$\arg\max_{k, k \leq 5, k \leq n}[\tilde{y}_{u,n-k} = \ldots = \tilde{y}_{u,n-1} = 0]$
backward skill continuity	$\arg\max_{k, k \leq 5, k+n \leq N_u}[\tilde{y}_{u,n+1} = \ldots = \tilde{y}_{u,n+k} = 0]$
mode ratio	$\frac{\sum_{n+i \in \text{window}} \mathbb{I}[\tilde{y}_{u,n+i}=1]}{2 \times sz}$
weighted mode ratio	$\sum_{-sz \leq i \leq sz, i \neq 0} \frac{1}{i} \times \mathbb{I}[\tilde{y}_{u,n+i} = 1]$
bias term	b

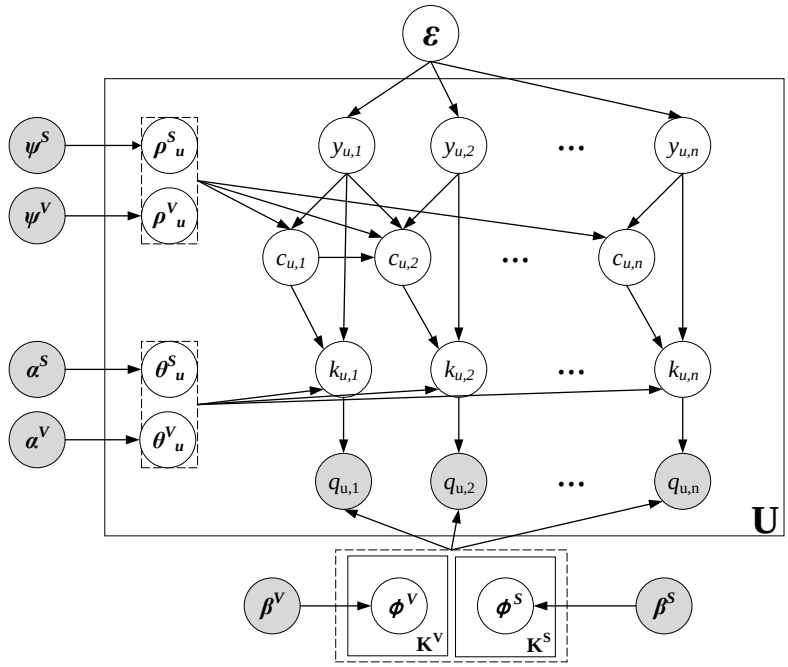


Fig. 6. The plate notation of the proposed model.

4.1.4 Enhancing the Mode Learning with Auxiliary Features. In the above, the mode preference η_u is drawn from a symmetric Beta prior. The learning of η_u fully relies on the sequential characteristics of the the trace data. Here, we further propose to incorporate useful auxiliary features for deriving the prior knowledge in determining the mode. Our idea is inspired by the observation in Fig. 5. Although it is difficult to identify the topic mode (since the topic itself is hidden), it is relatively easy to collect evidence that is useful to identify the volume mode. Intuitively, in a short window, if multiple attempted problems are from same volume and also have nearby problem IDs, it is probable to be in the volume mode. We compile a list of such auxiliary features, and derive a feature vector $\boldsymbol{x}_{u,n}$ for the n-th attempted problem by user u. We then train a binary logistic classifier to set the mode selection probability. The probability that selects the volume mode can be set below

$$\Pr(y_{u,n} = 1|\boldsymbol{x}_{u,n}, \boldsymbol{\epsilon}) = \frac{1}{1 + \exp(-\boldsymbol{\epsilon}^\top \cdot \boldsymbol{x}_{u,n})}, \tag{1}$$

where $\boldsymbol{\epsilon}$ is the weight vector for the logistic classifier. Once we have obtained the above conditional probabilities, we can then generate the mode by using η_u as the prior. Unlike η_u, $\Pr(y = 1|\boldsymbol{x}_{u,n}, \boldsymbol{\epsilon})$ is both user-specific and attempt-specific. Even for the same problem, when it appears in different sequential contexts, the mode selection probability will vary according to the actual context. For each attempt, we pre-compute a local volume mode indicator $\tilde{y}_{u,n}$. We set $\tilde{y}_{u,n}$ to 1, *iff* (1) $v_{u,n-1} = v_{u,n} = v_{u,n+1}$ and (2) the problem ID differences are smaller than a threshold of Δ, i.e., $|q_{u,n} - q_{u,n-1}| \leq \Delta$ and $|q_{u,n} - q_{u,n+1}| \leq \Delta$. We set Δ to 20. $\tilde{y}_{u,n}$ can be considered as a good surrogate of $y_{u,n}$ with evidence gathered for the volume mode. By incorporating the volume mode

1. For each skill topic $k = 1, ..., K^S$,
   (1) draw a multinomial distribution $\phi_k^S \sim Dir(\beta^S)$;
2. For each volume topic $v = 1, ..., K^V$,
   (1) draw a multinomial distribution $\phi_v^V \sim Dir(\beta^V)$, set $\phi_{v,q}^V = 0$, if $v_q \neq v$;
3. For each user $u = 1, ..., |\mathcal{U}|$,
   (1) draw a multinomial distribution $\theta_u^S \sim Dir(\alpha^S)$;
   (2) draw a multinomial distribution $\theta_u^V \sim Dir(\alpha^V)$;
   (3) draw a Bernoulli distribution $\rho_u^V \sim Beta(\psi^V)$;
   (4) draw a Bernoulli distribution $\rho_u^S \sim Beta(\psi^S)$;
   (5) For each problem $q_{u,n}(n = 1, ..., N_u)$ in the attemp sequence by u,
      (i.) draw the mode variable $y_{u,n} \sim \Pr(y|\boldsymbol{x}_{u,n}, \boldsymbol{\epsilon})$;
      (ii.) draw $c_{u,n}$ by,

$$c_{u,n} = \begin{cases} 1 & \text{if } y_{u,n} \neq y_{u,n-1}, \\ c \sim \text{Bern}(\rho_u^V) & \text{if } y_{u,n} = y_{u,n-1} = 1, \\ c \sim \text{Bern}(\rho_u^S) & \text{if } y_{u,n} = y_{u,n-1} = 0. \end{cases}$$

      (iii.) sample topic $k_{u,n}$ by,

$$k_{u,n} = \begin{cases} k_{u,n-1} & \text{if } c_{u,n} = 0, \\ v \sim Multi(\theta_u^V) & \text{if } y_{u,n} = 1, c_{u,n} = 1, \\ z \sim Multi(\theta_u^S) & \text{if } y_{u,n} = 0, c_{u,n} = 1. \end{cases}$$

      (iiii.) sample the problem $q_{u,n}$ by,

$$q_{u,n} = \begin{cases} q \sim Mul(\phi_v^V) & \text{if } y_{u,n} = 1, k_{u,n} = v, \\ q \sim Mul(\phi_z^S) & \text{if } y_{u,n} = 0, k_{u,n} = z. \end{cases}$$

Fig. 7. The generative process of the two-mode Markov topic model.

indicator $\tilde{y}_{u,n}$, we can derive a series of compositional features based on $\tilde{y}_{u,n}$ from a short window as summarized in Table 4.

4.1.5 Summary. We present the plate diagram of the proposed model in Fig. 6 and its full generative process in Fig. 7. Apart from hidden topic variables, we also include two binary hidden variables y and c to control the mode selection and mode continuity respectively. Furthermore, we introduce a general topic assignment $k_{u,n}$ for both skill and volume topics. Finally, a problem will be generated according to the corresponding topic model with the control of both y and c. We present the six generation routes for an attempted problem in Table 3. When $c = 1$, we need to sample a new topic assignment for the current problem according to the user-specific preference distribution as in standard LDA model [5]. The key in discriminating the two modes is the use of auxiliary features from local attempt context. Although we have two types of topics, namely volume topics and skill topics, the assignments of volume topics are actually observed. However, for consistency of derivation, we still treat the volume topic assignments as hidden, which are generated from a point mass distribution. That is, $\Pr(v = v_q|u, q) = 1$ and $\Pr(v = v'|u, q) = 0$ for all $v' \neq v_q$. Recall that our volume topics are truncated to the problems from the corresponding volume. In other words, $\Pr(q|v) = 0$ for any $v \neq v_q$. Based on this, $\Pr(q|v) \times \Pr(v|u) \neq 0$ iff when $v = v_q$.

4.1.6 Inference and Parameter Learning. In our model, the parameters to learn are topics $\{\phi_z^S\}$ and $\{\phi_z^V\}$, users' preferences $\{\theta_u^S\}$ and $\{\theta_u^V\}$, switch distributions $\{\rho_u^V\}$ and $\{\rho_u^S\}$, and the logistic weights $\boldsymbol{\epsilon}$. The hyper-parameters α, β, ψ are set to small values empirically. We need to infer the values of hidden variables, i.e., c, y and z.

Let Ψ denote all the hyper-parameters. We first derive the joint likelihood of all the variables (both hidden and observed) conditional on the hyper-parameters. For clarity of derivation, we do not expand the probability of $Pr(\phi_z^S, \phi_z^V, \theta_u^S, \theta_u^V, \rho_u^V, \rho_u^S, \boldsymbol{\epsilon}|\Psi)$. The joint likelihood is given below

$$\begin{aligned}
&Pr(\boldsymbol{k}, \boldsymbol{y}, \boldsymbol{c}, \boldsymbol{q}, \phi_z^S, \phi_z^V, \theta_u^S, \theta_u^V, \rho_u^V, \rho_u^S, \boldsymbol{\epsilon}|\Psi) \\
=&\prod_{u=1}^{|\mathcal{U}|}\prod_{n=1}^{N_u} Pr(y_{u,n}|u, \boldsymbol{\epsilon}) Pr(c_{u,n}|\rho_u, y_{u,n}, y_{u,n-1}) Pr(k_{u,n}|k_{u,n-1}, y_{u,n}, c_{u,n}, \theta_u^S, \theta_u^V) \\
&Pr(q_{u,n}|y_{u,n}, k_{u,n}, \phi_{k_{u,n}}^V, \phi_{k_{u,n}}^S) Pr(\phi_z^S, \phi_z^V, \theta_u^S, \theta_u^V, \rho_u^V, \rho_u^S, \boldsymbol{\epsilon}|\Psi),
\end{aligned} \tag{2}$$

where we use $k_{u,n}$ to denote either type of topics, and the rest factors can be expanded as follows

$$Pr(y_{u,n} = 1|\boldsymbol{x}_{u,n}, \boldsymbol{\epsilon}) = \frac{1}{1 + \exp(-\boldsymbol{\epsilon}^\top \cdot \boldsymbol{x}_{u,n})}. \tag{3}$$

$$Pr(c_{u,n}|\rho_u, y_{u,n}, y_{u,n-1}) = \begin{cases} 1 & \text{if } y_{u,n} \neq y_{u,n-1}, \\ \rho_u^V & \text{if } y_{u,n} = y_{u,n-1} = 1, \\ \rho_u^S & \text{if } y_{u,n} = y_{u,n-1} = 0. \end{cases} \tag{4}$$

$$Pr(k_{u,n} = k|k_{u,n-1}, y_{u,n}, \theta_u^S, \theta_u^V) = \begin{cases} \mathbb{I}[k_{u,n-1} = k] & \text{if } c_{u,n} = 0, \\ \theta_{u,k}^S & \text{if } y_{u,n} = 1, c_{u,n} = 1, \\ \theta_{u,k}^V & \text{if } y_{u,n} = 0, c_{u,n} = 1. \end{cases} \tag{5}$$

$$Pr(q_{u,n} = q|y_{u,n}, k_{u,n}, \phi_{k_{u,n}}^V, \phi_{k_{u,n}}^S) = \begin{cases} \phi_{z,q}^S & \text{if } y_{u,n} = 0, k_{u,n} = z, \\ \phi_{v,q}^V & \text{if } y_{u,n} = 1, k_{u,n} = v. \end{cases} \tag{6}$$

Our objective is to jointly optimize the above likelihood given the hyper-parameters. Since it involves both observed and hidden variables, it is computationally infeasible to perform direct optimization. In this work, we develop a method to perform approximate inference following the standard parameter estimation method for HMMs. Our model captures two types of sequential relatedness, namely topic-oriented and volume-oriented, in a chain-like structure. The user attempt sequence essentially forms a Markov chain. The topic assignment of an attempt will depend on its preceding attempt. Specially, in our model, each hidden triplet $\langle c_{u,n}, y_{u,n}, k_{u,n}\rangle$ stands for the state of the current attempt, which consists of three hidden assignments. So, the total number of hidden states in our model is $(2 \cdot |\mathcal{T}| + 2 \cdot |\mathcal{V}|)$. For each skill or volume topic, we associate it with two unique states to indicate whether it follows the previous assignment or is re-drawn. Following the standard Expectation-Maximization algorithm, we use the E-step is to infer the values for hidden variables, and the M-step to optimize the parameters. We omit the derivation details, and present the update formulas for the E-step and M-step.

In the E-step, we perform the calculation for the hidden variables

$$E(C_{z,q}) = \sum_{u=1}^{|\mathcal{U}|}\sum_{n=1}^{N_u} Pr(q_{u,n}=q, k_{u,n}=z, y_{u,n}=0|\boldsymbol{q}_u), \tag{7}$$

$$E(C_{v,q}) = \sum_{u=1}^{|\mathcal{U}|}\sum_{n=1}^{N_u} Pr(q_{u,n}=q, k_{u,n}=v, y_{u,n}=1|\boldsymbol{q}_u), \tag{8}$$

$$E(C_{u,z}) = \sum_{n=1}^{N_u} \Big\{ Pr(y_{u,n}=0, y_{u,n-1}=1, k_{u,n}=z|\boldsymbol{q}_u), \tag{9}$$

$$+Pr(y_{u,n}=0, y_{u,n-1}=0, c_{u,n}=1, k_{u,n}=z|\boldsymbol{q}_u)\Big\}, \tag{10}$$

$$E(C_{u,v}) = \sum_{n=1}^{N_u} \Big\{ Pr(y_{u,n}=1, y_{u,n-1}=0, k_{u,n}=v|\boldsymbol{q}_u), \tag{11}$$

$$+Pr(y_{u,n}=1, y_{u,n-1}=1, c_{u,n}=1, k_{u,n}=v|\boldsymbol{q}_u)\Big\}, \tag{12}$$

$$E(C_{u,n,i}) = Pr(y_{u,n}=1|\boldsymbol{q}_u), \tag{13}$$

$$E(C_{u,n,s}) = Pr(y_{u,n}=0|\boldsymbol{q}_u), \tag{14}$$

where $\boldsymbol{q}_u$ is the attempt sequence by excluding the n-th attempt.

While for the M-step, we can perform the following update to optimize the parameters

$$\phi^V_{v,q} \propto E(C_{v,q}) + \beta^V - 1, \tag{15}$$

$$\phi^S_{z,q} \propto E(C_{z,q}) + \beta^S - 1, \tag{16}$$

$$\theta^V_{u,v} \propto E(C_{u,v}) + \alpha^V - 1, \tag{17}$$

$$\theta^S_{u,z} \propto E(C_{u,z}) + \alpha^S - 1, \tag{18}$$

$$\rho^S_u = \frac{\sum_{n=2}^{N_u} Pr(y_{u,n}=0, y_{u,n-1}=0, c_{u,n}=1|\boldsymbol{q}_u) + \psi^S}{\sum_{n=2}^{N_u} Pr(y_{u,n}=0, y_{u,n-1}=0|\boldsymbol{q}_u) + 2\psi^S}, \tag{19}$$

$$\rho^V_u = \frac{\sum_{n=2}^{N_u} Pr(y_{u,n}=1, y_{u,n-1}=1, c_{u,n}=1|\boldsymbol{q}_u) + \psi^V}{\sum_{n=2}^{N_u} Pr(y_{u,n}=1, y_{u,n-1}=1|\boldsymbol{q}_u) + 2\psi^V}. \tag{20}$$

As we also have logistic parameters involved in the model, we need to learn the logistic weights $\boldsymbol{\epsilon}$ by maximizing the following objective function in the M-step.

$$\boldsymbol{\epsilon}^{j+1} = \underset{\boldsymbol{\epsilon}}{\operatorname{argmax}} \sum_{u=1}^{|\mathcal{U}|}\sum_{n=1}^{N_u} E(C_{u,n,i}) \log Pr(y_{u,n}=1|\boldsymbol{x}_{u,n}, \boldsymbol{\epsilon}) + E(C_{u,n,s}) \log Pr(y_{u,n}=0|\boldsymbol{x}_{u,n}, \boldsymbol{\epsilon}), \tag{21}$$

which can be optimized by using a stochastic gradient descent method

$$\frac{\partial J(\boldsymbol{\epsilon})}{\partial \epsilon_f} \tag{22}$$

$$= \sum_{u=1}^{|\mathcal{U}|}\sum_{n=1}^{N_u} x_{u,n,f} \cdot \left\{ E(C_{u,n,i}) \cdot \frac{\exp(-\sum_f \epsilon_f \cdot x_{u,n,f})}{1+\exp(-\sum_f \epsilon_f \cdot x_{u,n,f})} - E(C_{u,n,s}) \cdot \frac{1}{1+\exp(-\sum_f \epsilon_f \cdot x_{u,n,f})} \right\},$$

where ϵ_f is the weight of the f-feature.

In order to avoid local minima as commonly encountered when using the EM algorithm, we train the model with multiple random initializations.

4.2 Difficulty and Expertise Level Estimation using a Competition Model with Topical Contexts

In the above, we have studied how to learn topics using the two-mode Markov topic model. The proposed model is able to capture the sequential relatedness from two learning modes. Our final model outputs a set of skill topics (i.e., $\{\phi_z^S\}$), each of which groups problems with coherent skills under a specific topic. Now, we continue to study how to estimate user expertise and problem difficulty levels by introducing a novel competition model. Recall that we have created a virtual user for each problem (Section 2), we thus do not need to discriminate between user expertise and problem difficulty unless otherwise specified. Our focus is to predict (1) whether a user u is qualified to solve a problem q; and (2) whether is a user u is more capable of solving a problem q than another user u'.

4.2.1 The Bradley-Terry comparison model. The estimation for user expertise and problem difficulty levels has been widely studied in the past [13, 21]. Most of existing studies assume that each user or problem is associated with a single expertise or difficulty score. Our base model is built on the classic Bradley-Terry model [47], which is the basis of many research works in pairwise comparison. In the Bradley-Terry model, each player's strength is represented by a single real number, and the probability of player u beating player u' is modeled as

$$\begin{aligned} \Pr(u \text{ beats } u') &= \frac{\exp(e_u)}{\exp(e_u) + \exp(e_{u'})}, \\ &= \frac{1}{1 + \exp(-(e_u - e_{u'}))}, \\ &= S(e_u, e_{u'}), \end{aligned} \tag{23}$$

where e_u and $e_{u'}$ are the expertise scores of user u and u' respectively. The Bradley-Terry model essentially adopts the sigmoid function which takes the difference between the strengths of the two players. The actual "competition" case can be very complex, and a single expertise score has a relatively weak capability to measure the expertise level of a user or a problem in varying contexts. Hence, we consider incorporating the contextual information for two-player comparison into the Bradley-Terry model. Given the ECT $\langle u, u', q\rangle$, a problem has to be specified as the context of a competition, which is defined below

$$\begin{aligned} \Pr(u \text{ beats } u'|q) &= \frac{\exp(e_u^{(q)})}{\exp(e_u^{(q)}) + \exp(e_{u'}^{(q)})}, \\ &= \frac{1}{1 + \exp\left(-(e_u^{(q)} - e_{u'}^{(q)})\right)}, \\ &= S(e_u^{(q)}, e_{u'}^{(q)}), \end{aligned} \tag{24}$$

where $e_u^{(q)}$ is the expertise score that u attempts on q. In other words, $e_u^{(q)}$ is the expertise score of user u with the context of q. As shown in *Obs.* 3, it is easy to see that the topic information of a problem is likely to influence its difficulty level. We would like to characterize the topic information using the extracted skill topics.

4.2.2 Expertise Modeling with Topical Contexts. In order to define $e_u^{(q)}$, we have to consider two points: (1) how to represent the contextual information from problem q; and (2) how to define the context-aware expertise score $e_u^{(q)}$. First, we extract the skill topic information related to a problem and define the context as the distribution over the skill topics. Formally, for each problem q, we can obtain its conditional probabilities in all the skill topics, i.e., $\{\phi^S_{k,q}\}_k$. Note that these conditional probabilities $\{\phi^S_{k,q}\}_k$ for problem q do not sum to 1. For simplification of notations, we use $\boldsymbol{x}_q$ to denote the K^S-dimensional vector consisting of the probability set $\{\phi^S_{k,q}\}_k$ for problem q. In this way, the contexts from all the problems can be represented in a unified way. After introducing topical contexts, we can further define the topic-aware expertise. Formally, we assume that a user u is associated with a base expertise score b_u, which is static and fixed. In addition, let $\boldsymbol{s}_u \in \mathbb{R}^{K^S}$ denote a topic-specific expertise vector for user u, where each entry $s_{u,k}$ denotes the expertise level of user u on the k-th topic. With the above definitions, we can define the expertise score that user u attempts to solve problem q as follows

$$e_u^{(q)} = \underbrace{\boldsymbol{w}^\top \cdot \left(\boldsymbol{x}_q \odot \boldsymbol{s}_u\right)}_{\text{problem-specific expertise}} + \underbrace{b_u}_{\text{base expertise}}, \tag{25}$$

where $\boldsymbol{w} \in \mathbb{R}^{K^S}$ is a weight vector which measures the global importance for different topics, $\boldsymbol{x}_q \in \mathbb{R}^{K^S}$ is a contextual vector with topical information for problem q, and " $\odot$ " denote the element-wise product for vectors. By incorporating the problem as context, the topical vector $\boldsymbol{x}_q$ weights the topic-specific expertise vector $\boldsymbol{s}_u$ in different topic dimensions. In this way, $e_u^{(q)}$ is decomposed into two parts, namely base expertise score and problem-specific expertise score.

We further put prior on the model parameters. The base score is drawn from a user-specific Gaussian distribution, i.e.,

$$b_u \sim \mathcal{N}(\mu_u, \sigma_u), \tag{26}$$

where μ_u and σ_u are the prior parameters of a Gaussian distribution, which are user-specific. Similarly, the topic-specific expertise vector can be drawn as follows

$$\boldsymbol{s}_u \sim \mathcal{N}(\boldsymbol{\mu}_u, \Sigma_u), \tag{27}$$

where $\boldsymbol{\mu}_u = [\mu_u, ..., \mu_u]^\top$ is a K^S-dimensional vector of μ_us, and $\Sigma_u = \sigma_u \cdot \boldsymbol{I}$ is the diagonal matrix of size $K^S \times K^S$. In the above, the user-specific prior parameters (μ_u, σ_u) essentially control both b_u and $\boldsymbol{s}_u$. We can put hyper-prior on (μ_u, σ_u). However, such a full Bayesian approach will make the optimization too complicated. Hence, we pre-train the classic TrueSkill model [28] to set the (μ_u, σ_u). The TrueSkill is a strong skill-based ranking system, in which a player's skill is represented as a normal distribution characterized by a mean value and a variance. We use the learned means and variances to set the (μ_u, σ_u) in our model. By taking the output of TrueSkill as prior parameters, we can obtain a high-quality prior and better guide the learning of the parameters.

Finally, we put prior on $\boldsymbol{w}$, which is drawn from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The overall model description is summarized in Fig. 8.

1. Draw a global topic importance vector $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$;
2. For each user $u = 1, ..., |\mathcal{U}|$
   (1) set μ_u and σ_u using TrueSkill;
   (2) draw the base expertise score $b_u \sim \mathcal{N}(\mu_u, \sigma_u)$;
   (3) draw the topic-specific expertise vector $\boldsymbol{s}_u \sim \mathcal{N}(\boldsymbol{\mu_u}, \sigma_u \boldsymbol{I})$;
   (4) derive the context-aware expertise score $e_u^{(q)}$ using Eq. 25;
3. For each problem $q = 1, ..., |\mathcal{Q}|$,
   (1) derive $\boldsymbol{x}_q$ using the topic model in Section 4.1;
4. For each ECT $\langle u, u', q \rangle$ in the dataset,
   i. draw the competition result using Eq. 24.

Fig. 8. The description of the topic-based competition model.

4.2.3 Optimization. After defining $e_u^{(q)}$, we can plug Eq. 25 into Eq. 24 and derive the probability for a ECT $\langle u, u', q \rangle$, which indicates that u beats u' on the problem q. In other words, u successfully solved q but u' failed in solving q. We present the overall objective function to learn the expertise score using the training ECT set $\mathcal{D}$ as below

$$\mathcal{L}(\mathcal{G}) = \sum_{\langle u, u', q \rangle \in \mathcal{D}} \log \Pr(u \text{ beats } u' | q, \Phi) + \lambda \log \Pr(\Phi), \tag{28}$$

where the loss function is essentially a Bayesian decomposition in a log form, and Φ denote all the related parameter, including $\boldsymbol{w}$, base scores $\{b_u\}$ and topic-specific expertise vector $\{\boldsymbol{s}_u\}$.

Note that we have created a virtual user for each problem. We treat the related expertise score from each virtual user as the difficulty level of its corresponding problem. To generate the training ECT set $\{\langle u, u', q \rangle\}$, we need to consider two types of comparisons: (1) a user *v.s.* a problem, and (2) a user *v.s.* another user. The first type is achieved by mapping problems to virtual users, while the second type can be constructed by selecting intermediate problems. Once the triplets have been generated, we can adopt the standard Stochastic Gradient Descent approach for parameter optimization. At each time step, we sample a mini batch of 128 ECTs and then update the parameters involved in these ECTs.

4.3 Applications of the Proposed Approaches to Online Judge Systems

In this section, we discuss how to apply the proposed approaches to improve OJ systems.

4.3.1 Topic Extraction and Problem Classification. The learned skill topics (e.g., *dynamic programming*, *backtracking*, etc.) using our proposed two-mode Markov topic model can be helpful to organize problems in a topical way. Here, we consider a practical application, i.e., how to perform problem classification based on the learned skill topics. Given a set of predefined categories $\mathcal{A}$, we would like to classify a problem q using very little labeled data $\{(q', a_{q'}) | a_{q'} \in \mathcal{A}\}$. We classify a problem based on the following equation:

$$
\begin{aligned}
a_q &= \operatorname*{argmax}_{a \in \mathcal{A}} \Pr(q|a), \\
&= \operatorname*{argmax}_{a \in \mathcal{A}} \prod_{a_{q'}=a} \Pr(q|q'), \\
&= \operatorname*{argmax}_{a \in \mathcal{A}} \prod_{a_{q'}=a} \sum_{z \in \mathcal{T}} \Pr(q|z) \times \Pr(z|q'), \\
&\propto \operatorname*{argmax}_{a \in \mathcal{A}} \prod_{a_{q'}=a} \sum_{z \in \mathcal{T}} \Pr(q|z) \times \Pr(q'|z) \times \Pr(z),
\end{aligned} \tag{29}
$$

where z is a skill topic, and the involved three factors $\Pr(q|z), \Pr(q'|z), \Pr(z)$ can be directly obtained using our model. We assume that the number of labeled problems is the same for each category.

4.3.2 Expertise Level Estimation and Competition Prediction. Our expertise model can generate expertise scores for both users and problems. Since we create a virtual user for each problem, problem difficulty and user expertise can be measured in the same scale. Following Eq. 24, we can predict the outcome of (1) an attempt of a user on a problem; or (2) a competition between two users on a problem.

4.3.3 Problem Recommendation. By combining the above two aspects, the third application aims to recommend problems to a user by considering both aspects of *topical interests* and *expertise levels*. Intuitively, a user has a topical preference over the problems, and would attempt the problems that have matched her topical interests. Also, a user cannot solve problems beyond her expertise level. We can therefore make recommendations of problems to users by combining these two factors.

Specially, we consider two scenarios for recommendation. The first scenario is overall recommendation, which aims to generate a list of time-insensitive recommendations, while the second scenario is the next-basket recommendation, which predicts the problems that a user is going to attempt next. Formally, the overall recommendation task can be casted as a similarity evaluation problem between a user u and q as follows

$$
score_{overall}(u,q) = \underbrace{\left\{ \sum_{v} \phi^{V}_{v,q} \cdot \theta^{V}_{u,v} \cdot \eta_u + \sum_{z} \phi^{S}_{z,q} \cdot \theta^{S}_{u,z} \cdot (1-\eta_u) \right\}}_{\text{Interst}} \times \underbrace{\Pr(u \text{ beats } u_q|q)}_{\text{Expertise}}, \tag{30}
$$

where $\Pr(u \text{ beats } u_q|q)$ is defined in Eq. 24. Our scoring function considers both topical interests and expertise scores. For topical interest, we further calculate two preference scores for skill and volume topics respectively.

For next-basket recommendation, we make the prediction based on the users' preference and the previous attempt history

$$
\begin{aligned}
& score_{next}(u,q) \qquad (31)\\
= \; & \underbrace{\Pr(q_{u,n}=q|u,q_{u,1},\cdots,q_{u,n-1},\Phi)}_{\text{Interst}} \times \underbrace{\Pr(u \text{ beats } u_q|q)}_{\text{Expertise}}\\
= \; & \Pr(y_{u,n}|u,\boldsymbol{\epsilon})\\
& \times \Pr(c_{u,n}|\rho_u, y_{u,n}, y_{u,n-1})\\
& \times \sum_{z=1}^{K} \Pr(k_{u,n}=z|k_{u,n-1}, y_{u,n}, c_{u,n}, \theta_u^S, \theta_u^V)\\
& \times \Pr(q_{u,n}|y_{u,n}, k_{u,n}, \phi_{k_{u,n}}^V, \phi_{k_{u,n}}^S)\\
& \times \Pr(u \text{ beats } u_q|q),
\end{aligned}
$$

where the first four terms are defined in Eq. 3 to Eq. 6 respectively, and $\Pr(u \text{ beats } u_q|q)$ is defined in Eq. 24.

Different from conventional recommendation algorithms, we incorporate expertise into consideration for recommendation. We decrease the score of a candidate problem which matches the interests of a user well but exceeds her expertise level. The learned topic information is mainly used for interest-driven recommendation, while the learned expertise/difficulty information is mainly used for expertise-based recommendation. The final recommendation is a balanced trade-off between these two factors.

5 EXPERIMENTS

In this section, we conduct experiments on the three datasets presented in Table 1 to evaluate our proposed models on the three application scenarios mentioned in Section 4.3, namely, skill topic extraction, expertise competition prediction and problem recommendation.

5.1 Skill Topic Extraction

Automatic learning of skill topics for problems is extremely important for OJ systems since the learned topics can be used to group problems which makes it more easier for users to search for problems under certain topics.

5.1.1 Experimental setup. Recall that for each of the three datasets, we have the topic or category labels for a fraction of problems. Hence, our experiments are evaluated only using the partial set of labeled problems. We evaluate our approach in the following aspects, namely, topic similarity, topic purity and problem classification.

- *Topic similarity*: Given a problem q, we can obtain a vector $\{\phi_{z,q}\}_{z\in\mathcal{T}}$ consisting of its probabilities generated by each topic, where each entry $\phi_{z,q}$ is the probability of problem q generated by topic z. The intuition is that if two problems are from the same category, then their topic-probability vectors should be also similar. Based on this idea, given two categories, we compute the average similarity between the learned topic-probability vectors for the problems in the two categories. Formally, let C_1 and C_2 denote two labeled problem sets, each of which corresponds to a category. We compute the pairwise similarities between two categories as follows

$$
sim(C_1, C_2) = \frac{1}{|C_1| * |C_2|} \sum_{q\in C_1} \sum_{q'\in C_2} \frac{\sum_{z\in\mathcal{T}} \phi_{z,q} \cdot \phi_{z,q'}}{\| \phi_{\cdot,q} \|_2 \cdot \| \phi_{\cdot,q'} \|_2}, \qquad (32)
$$

where we use the cosine similarity to measure the similarity between two topic-probability vectors.

- *Topic Purity*: Topic models can be considered as a soft clustering of items. Hence, we measure the quality of the generated skill topics using *purity*, a commonly used metric for clustering evaluation. Given a topic, we take the top ten problems to form a cluster. Then we compute the purity score for a set of topics as follows

$$Purity(\mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{z \in \mathcal{T}} \frac{N^{(z)}_{a_{max}}}{\sum_{a'} N^{(z)}_{a'}}, \tag{33}$$

where $N^{(z)}_a$ denote the number of problems with the category label a and a_{max} is the most common category label in topic z.

- *Problem classification*: The eventual goal of learning skill topics is to classify problems into categories. We follow the derivations in Section 4.3 to classify a problems as follows

$$a_q \propto \arg\max_{a \in \mathcal{A}} \prod_{a_{q'}=a} \sum_{z \in \mathcal{T}} \Pr(q|z) \times \Pr(q'|z) \times \Pr(z), \tag{34}$$

where we only use five labeled problems for each category. Since it is a multi-classification task, we use the F1 and accuracy as the evaluation metrics.

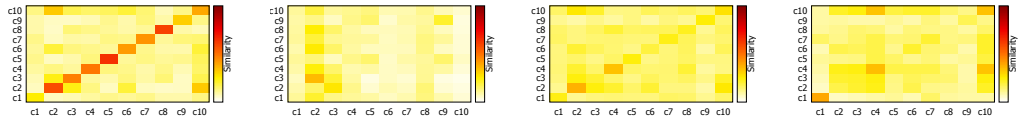
5.1.2 Methods to Compare. We compare our model with three topic models and a Hidden Markov Model.

- *LDA*: We use the Latent Dirichlet Allocation [5] to generate skill topics. We first combine all the attempted problems by a user as a document, then run the standard LDA model using GibbsLDA++ [41]. LDA does not consider the order of word tokens, and it cannot discriminate between skill topics and volume topics. We apply the default settings in [22], i.e., $\alpha = 50/K, \beta = 0.01$.
- *LDA$_{text}$*: Instead of using problems as word tokens, we directly crawl the textual problem descriptions from OJ websites. Then we remove the stopwords and run the standard topic extraction using the LDA model. Note that in this method, we cannot learn problem topics but instead word topics. Hence, we only use LDA$_{text}$ in the evaluation of problem classification. The hyper-parameters are set as $\alpha = 50/K, \beta = 0.01$.
- *HTMM*: It is the Hidden Markov Topic Model [24], in which the order of tokens has been characterized based on the first-order Markovian property. In our setting here, the attempted problem sequence from a user is considered as a document. The hyper-parameters are set as $\alpha = 1 + 50/K, \eta = 1.01$.
- *HMM*: It is the standard first-order Hidden Markov Model, which is used to characterize the attempted sequences. For fairness, we implement the HMM model using Gibbs sampling, where the state hidden variables can be considered as problem topics.
- *Our model*: It is our proposed model in Eq. 29. We only use the learned skill topics for evaluation. The hyper-parameters are set as follows: $\alpha^V = \alpha^S = 1 + 50/K, \beta^V = \beta^S = 1.01, \psi = 0.01$.

We set the number of topics to 40 for all the compared methods based on a validation set of 10% training data.

5.1.3 Results and Analysis. We first present the results of *topic similarity* on the Timus dataset in Figure 9. We use the heatmaps to visualize the pairwise similarities between two categories. It can be observed that our model produces a clear diagonal lines consisting of large similarity values,

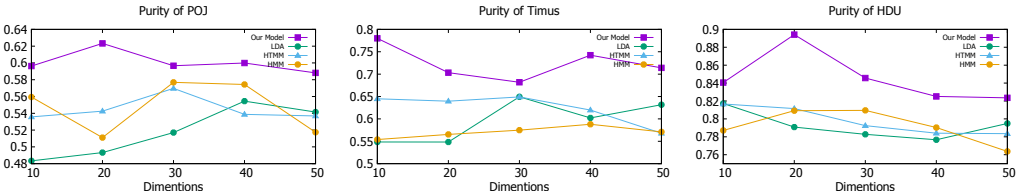
while the other three methods have even distributions for category similarities. The results indicate that our generated skill topics are very effective to discriminate problems from different categories. We also find hat it is not easy to learn clear skill topics using traditional models like LDA and HMM. One main reason is that the generation of attempt sequences is a mixture of two learning modes. while the baselines can only consider one of the two modes.



(a) Our model (b) $HTMM$ (c) HMM (d) LDA

Fig. 9. Heatmaps for pairwise category similarity measured by different topic models on the Timus dataset.

Next, we present the purity scores of different methods on the three datasets in Figure 10. We vary the number of topics (or states in HMM) from 10 to 50 with a step of 10. It can be observed that our model is consistently better than the baselines in all the cases. For our model, a topic number of 20 gives best performance on the datasets of POJ and HDU, while a topic number of 10 gives best performance on the dataset of Timus. The main reason is that Timus has fewer problems than the other two OJ websites.



(a) POJ (b) Timus (c) HDU

Fig. 10. Purity scores of different methods on three datasets.

We finally present the results of problem classification in Table 5. We can make the following observations. First, the text-based LDA_{text} model is much worse than the problem-based LDA model, which indicates that the textual content is less useful for inferring the problem topics. Our result is consistent with the finding in [50], where it has shown that often times the problem statements are obscure for determining the topic information. Overall, the baseline HTMM performs the best among all the baselines, since it considers the sequential relatedness. Our model significantly improves over all the baselines with a large margin, which demonstrates the effectiveness of our model on problem classification.

5.1.4 Effect of different mode features. In our model, we incorporate a binary logistic classifier component for predicting the learning mode $y_{u,n}$ for the n-th attempted problem from a user u based on a set of features. When $y_{u,n} = 1$, it is the volume mode, and when $y_{u,n} = 0$, it is the topic mode. Our features mainly consist of the features derived from local volume mode indictors $\tilde{y}_{u,n}$ in a window with the size set to five (See Section 4.1.4). We report the learned feature weights of our

Table 5. Performance comparison for problem classification on three datasets.

Methods	*POJ*		*Timus*		*HDU*	
	F1	Accuracy	F1	Accuracy	F1	Accuracy
LDA	0.659	0.713	0.464	0.49	0.564	0.596
LDA_{text}	0.382	0.413	0.341	0.392	0.387	0.396
HMM	0.605	0.647	0.568	0.596	0.546	0.592
HTMM	0.627	0.667	0.473	0.493	0.561	0.613
Our model	**0.751**	**0.817**	**0.644**	**0.707**	**0.724**	**0.788**

model on the Timus dataset in Table 6. It is clear to see that the weights of volume-related features are positive while the weights of skill-related features are negative, which confirms to our intuition.

Table 6. Effect analysis of the weights for different features. $\mathbb{I}[\cdot]$ is an indicator function which returns 1 *iff* when the condition is true.

Mode Feature	Formulations	Weight
local volume continuity	$\mathbb{I}[\tilde{y}_{u,n-1} = 1 \text{ and } \tilde{y}_{u,n+1} = 1]$	1.243
local skill continuity	$\mathbb{I}[\tilde{y}_{u,n-1} = 0 \text{ and } \tilde{y}_{u,n+1} = 0]$	-3.186
forward volume continuity	$\arg\max_{k,k\leq 5,k\leq n}[\tilde{y}_{u,n-k} = \ldots = \tilde{y}_{u,n-1} = 1]$	1.261
backward volume continuity	$\arg\max_{k,k\leq 5,k+n\leq N_u}[\tilde{y}_{u,n+1} = \ldots = \tilde{y}_{u,n+k} = 1]$	1.261
forward skill continuity	$\arg\max_{k,k\leq 5,k\leq n}[\tilde{y}_{u,n-k} = \ldots = \tilde{y}_{u,n-1} = 0]$	-0.201
backward skill continuity	$\arg\max_{k,k\leq 5,k+n\leq N_u}[\tilde{y}_{u,n+1} = \ldots = \tilde{y}_{u,n+k} = 0]$	-0.775
mode ratio	$\frac{\sum_{n+i\in\text{window}}\mathbb{I}[\tilde{y}_{u,n+i}=1]}{2\times sz}$	0.02
weighted mode ratio	$\sum_{-a\leq i\leq sz, i\neq 0}\frac{1}{i}\times\mathbb{I}[\tilde{y}_{u,n+i} = 1]$	0.21
bias term	b	1.519

Summary. Our model can effectively discriminate between the two learning modes, and generate clear skill topics. The learned skill topics enhance the within-category similarity and reduce the cross-category similarity. These topics have a higher purity score than the baselines. Moreover, they are helpful to improve the performance of problem classification.

5.2 Evaluation on Expertise Competition Prediction

Problem difficulty and user expertise estimation has been a hot research topic in educational data mining [11, 42, 64]. It is particularly useful to help schedule the curriculum and practice for students. In this part, we examine the performance of the proposed model on user expertise estimation.

5.2.1 Experimental Setup. In our task, it is infeasible to assign an absolute score to measure the difficulty level of a problem or the expertise of a user. Following the idea in ranking-based competition systems [28], we propose to use the competition based comparisons for evaluating the performance of various methods. Specially, we consider two kinds of competition scenarios, i.e., a user *v.s.* a problem and a user *v.s.* another user.

We first generate the labeled competitions between a user and a problem. Given a user u and an attempted problem q, there can be two outcomes, either accepted or non-accepted. We use $UQ-AC$ and $UQ-NAC$ to denote these two types. Next, we generate the labeled competitions between two users. Sometimes, it might be difficult to make a direct comparison between two users since they may have expertise in two different areas. For example, a user may be good at dynamic

Table 7. Statistics of the three competition types on three datasets.

Datasets	#UQ-AC	# UQ-NAC	# UUQ
Timus	1, 004, 817	118, 396	966, 438
POJ	2, 897, 504	193, 960	5, 213, 181
HDU	2, 473, 997	121, 157	3, 550, 390

programming while another may be good at graph algorithms. Hence, it would be sensible to include a problem as the context. In this case, we essentially predict the outcome for the competition triple $\langle u, u', q\rangle$, i.e., whether user u is more capable than user u' for problem q. To generate such labeled triplets, we start by selecting a difficult problem, e.g., the problems with fewer than 500 accepted solutions. Given a selected problem, we further select a user pair, in which a user has solved this problem while the other user has failed in this problem. We denote the third competition type by UUQ. In this way, we can generate a large amount of labeled ECTs. We combine the labeled competitions for different types of competition scenarios and form a final evaluation set. We present the statistics of the thee competition types in Table 7.

Since our datasets are imbalanced as there are more $UQ - AC$ competitions than $UQ - NAC$ competitions, we perform oversampling to duplicate $UQ - NAC$ competitions by five times. We then randomly split the labeled competitions with a ratio of 4:1 into training and test sets. For the training set, we take 10% to form the development set for optimizing model parameters. In our experiments, accuracy is used as the evaluation metrics, which is defined as follows:

$$ACC = \frac{\#correctly\_predicted\_competitions}{\#all\_competitions}. \tag{35}$$

5.2.2 Methods to Compare. The methods to be evaluated in our data are listed below:

- *TrueSkill*: The TrueSkill ranking system [28] is a skill based ranking system for Xbox Live developed at Microsoft Research. For each player, it characterizes her average skill and the degree of uncertainty in the player's skill. We use the open source implementation from http://trueskill.org.
- *Bradley-Terry*: In the Bradley-Terry model, each player u's strength is represented by a single real number e_u, and the probability of player u beating player u' is modeled as

$$\Pr(u \text{ beats } u') = \frac{1}{1 + \exp(-(e_u - e_{u'}))}. \tag{36}$$

- *PageRank*: We follow [33] to build a competition graph consisting of users and problems as vertices. Then we run the standard PageRank algorithm on the competition graph. The final competition results are derived by comparing the PageRank scores.
- *Blade-chest*: Unlike standard preference-learning models that represent the properties of each item/player as a single number, the blade-chest method [8] infers a multi-dimensional representation for the different aspects of each item/player's strength. We use the open-source implementation from https://github.com/csinpi/blade_chest.
- *AC rank*: We simply rank users by their number of solved problems. Note the AC rank method can only predict the competition result between two users.
- *Our model*: It is our proposed model in Eq. 24, in which we use the topic-related information from problems as the context for competition. Similar to the blade-chest model, we characterize the skill-specific expertise by a multi-dimensional representation. Note that we also

incorporate a base expertise in the proposed model. In addition, we consider two variants by removing the skill-specific expertise or base expertise, denoted by *our model*$_{\neg skill}$ and *our model*$_{\neg base}$ respectively. We set the number of topics K to 40.

It is worth noting that all the baselines cannot utilize the problem information in the UUQ competitions, since they can only characterize pairwise user expertise comparison without modelling problem context.

5.2.3 Results and Analysis. We report the performance of competitions in Table 8. We can make the following observations.

- TrueSkill performs very well for the competition type $UQ-AC$, but poorly for $UQ-NAC$. One main reason we have found is that TrueSkill tends to assign a higher expertise score to a user than a problem in the user-problem competition, since the majority of the attempted problems are essentially solved by users (See Table 7). Although oversampling can alleviate this problem to some extent, TrueSkill is still sensitive to data imbalance.
- Compared with TrueSkill, Bradley-Terry and PageRank performs worse on $UQ-AC$ but better for the other two types. We have empirically found that Bradley-Terry and PageRank seem to be more robust to deal with imbalanced training data.
- The simple baseline AC rank works well for predicting UUQ competitions. The AC rank method is effective to discriminate between a skilled user and a new user. And our UUQ competitions are created by first selecting difficult problems, which tends to generate more comparisons between users with different expertise levels.
- Blade-chest performs the best among all the baselines. It uses a multi-dimensional representation to characterize a player's skill in multiple latent dimensions, which tends to represent users' expertise more accurately.
- Our model is consistently better than all the baselines in all three competition types. The improvement over TrueSkill on $UQ-AC$ is relatively small, but the improvement on the other two types is significant. We also report the performance from the variants by removing skill-specific or base expertise. As we can see, the incorporation of skill-specific expertise is the key to yield the major improvement.

Table 8. Performance comparison for expertise competition prediction on three datasets.

Methods	*POJ*			*Timus*			*HDU*		
	UQ-AC	UQ-NAC	UUQ	UQ-AC	UQ-NAC	UUQ	UQ-AC	UQ-NAC	UUQ
TrueSkill	0.922	0.645	0.831	0.942	0.635	0.826	**0.945**	0.689	0.828
Bradley-Terry	0.899	0.711	0.844	0.909	0.823	0.832	0.922	0.803	0.833
PageRank	0.901	0.793	0.834	0.912	0.831	0.855	0.902	0.815	0.831
Blade-chest	0.885	0.824	0.875	0.858	0.844	0.866	0.889	0.814	0.863
AC rank	—	—	0.853	—	—	0.857	—	—	0.846
our model$_{\neg base}$	0.924	0.851	0.870	0.942	0.872	0.888	0.939	0.876	0.881
our model$_{\neg skill}$	0.918	0.847	0.865	0.923	0.865	0.887	0.932	0.865	0.878
our model	**0.929**	**0.868**	**0.884**	**0.944**	**0.874**	**0.897**	0.943	**0.881**	**0.886**

5.3 Evaluation on Problem Recommendation

We finally consider the evaluation on problem recommendation. Question recommendation can be utilized to improve OJ systems by recommending appropriate problems to users which could significantly reduce users' search efforts and improve user experience.

5.3.1 Experimental Setup. To make our evaluation more practical, we consider two recommendation scenarios. The first one is overall recommendation, which aims to recommend a list of problems based on the problems already attempted or solved by a user. The second one is next-basket recommendation for predicting the problems that a user is going to attempt in the next time period.

For overall recommendation, we randomly split the attempted or solved problems by a user with a ratio of 4:1 into training and test data. Similar to Information Retrieval, we use $Precision@k$ and $Recall@k$ as the evaluation metrics.

For next-basket recommendation, we hold out the last ten attempted or solved problems by a user. Then we predict the held-out problems for a user in a sequential manner. In each time period, we assume that all the previous problems have already been observed, and we predict what the next problem is going to be for a user. Following [60], we adopt $hit@k$ as the evaluation metrics, which calculates the ratio of the correct predictions in the top ten recommendations for all the users.

For both tasks, we set k to 10. Specifically, for both recommendation scenarios, we can also consider either the attempted problems or the solved problems as the ground truth. The attempted problems reflect users' interest, while the solved problems reflect both interest and expertise for users. We learn the models on the training data and examine its performance on the test data. We take 10% users from the training data as the development set to optimize model parameters.

5.3.2 Methods to Compare. We first present the methods to compare for overall recommendation.

- *UserKNN*: It is a classic collaborative filtering algorithm, which make the recommendations based on the top similar neighbors. We apply the Jarccard coefficient to find the most similar neighbors for a user based on her historical data. The number of similar neighbors is set to 50.
- *PMF*: It is the commonly used Probabilistic Matrix Factorization [37] model. We run the standard implementation of PMF on user-item interaction matrix. The number of latent factors is set to 40.
- *BPR*: It is the Bayesian Personalized Ranking model [48], which proposes a generic optimization criterion for personalized ranking that is the maximum posterior estimator derived from a Bayesian analysis. The number of latent dimensions is set to 40. BPR is a widely used baseline for recommendation with implicit feedback.
- *NCF*: It is the Neural Collaborative Filtering model [27], which utilizes deep learning techniques to enhance the modeling of user-item interactions. We follow the optimal setting tuned by [27] with three hidden layers. We also pretrain the NCF model as suggested in [27], which is important to improve its performance.
- *Our model*: It is our proposed model, which ranks the problems according to Eq. 30. We also consider two variants of our model, which either removes interest modeling or expertise modeling. We denote the two variants by *our model*$_{\neg int}$ and *our model*$_{\neg exp}$. The number of topics is set to 40.

Next, we present the methods to compare for next-basket recommendation.

- *TOP*: It is the frequency based recommendation method as mentioned before.
- *MC*: It is the frequency based Markov Chain model, which estimates the transition probability between two problems using the maximum likelihood estimation method.
- *FPMC*: It is the Factorized Personalized Markov Chain (FPMC) model [49], which subsumes both a common Markov chain and the normal matrix factorization model. It is an extension

of the BPR model for the next-basket recommendation. The number of latent factors is set to 40.

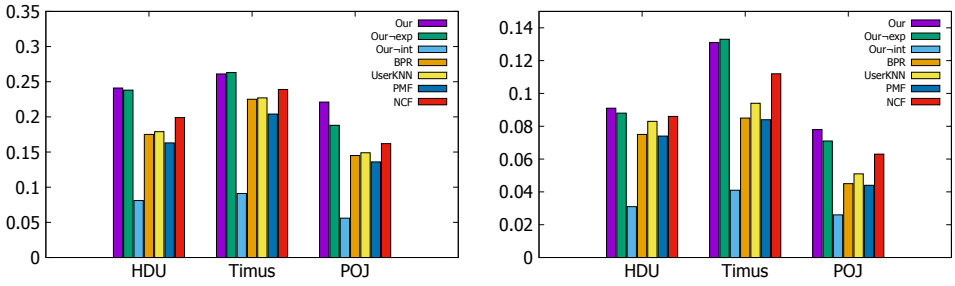
- *HRM*: It is the Hierarchical Representation Model (HRM) [60] for next-basket recommendation. HRM can capture both sequential behavior and users' general taste by involving transaction and user representations in prediction with distributed representations. The number of embedding dimensions is set to 40.
- *Our model*: It is our proposed model, which ranks the problems according to Eq. 31. We also consider two variants, which either removes interest modeling or expertise modeling. We denote the two variants by *our model*$_{\neg int}$ and *our model*$_{\neg exp}$. The number of topics is set to 40.

5.3.3 Results and Analysis. We first present the results for overall recommendation in Figure 11 and 12, which consider the attempted problems or the solved problems for recommendation respectively. We can make the following observations:

- The two baselines PMF and BPR give similar performance on the three datasets. Both methods adopt the latent representations to characterize the user preference. The classic UserKNN method performs very well and even slightly outperforms PMF and BPR. One main reason is that OJ systems are an open competitive platform, where a user can trace all the attempt records of others. In this way, users are likely to follow the attempt traces of those who have a similar expertise level, which makes similar users become more similar.
- The NCF model performs best among all the baseline methods. It utilizes a multi-layer perceptron component to characterize the complex user-item interactions, which is ideally to be able to model arbitrary binary interaction relations.
- Our proposed model is consistently better than the baselines. An interesting observation is that the improvement over the baselines in the solved-problem recommendation is more significant than that in the attempted-problem recommendation. Attempted-problems mainly reflect users' interest, while solved-problems reflect both users' interest and expertise. Hence, our model can gain more improvement for the recommendation of solved-problem recommendation by modeling both interest and expertise.
- By looking into Figure 11, we can see that the variant by excluding the expertise part is only slightly worse than the complete model with both parts, i.e., *our model*$_{\neg exp} \approx$ *our model*, in recommendation of attempted questions. However, as shown in Figure 12, the difference between these two variants become more significant in the recommendation for solved problems. In practical applications, we should recommend problems which match both users' interest and expertise.

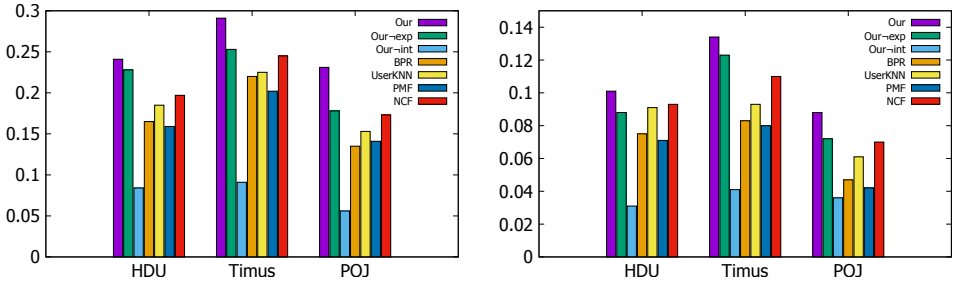
Next, we continue to examine the performance of different methods for next-basket recommendation. We present the results in Table 9. We can make the following observations:

- For the baselines, we have the performance ranking as: $HRM > FPMC > MC > TOP$. The simple baseline TOP performs the worst and MC improves over TOP in most cases, especially on the dataset of Timus. This is because Timus has fewer problems, in which the frequency-based method can achieve more reliable estimations.
- FPMC further improves over MC by modeling the transitions using matrix factorization, which can alleviate the problem of data sparsity for estimation. The HRM model performs the best among all the baselines, which applies the distributed representations to solve next-basket recommendation.
- Our proposed model is consistently better than the baselines, which characterizes the sequential recommendation with a two-mode topic modeling approach. One key merit



(a) Precision@10 (b) Recall@10

Fig. 11. Performance comparison for overall recommendation with *the attempted problems* as ground truth on three datasets. The improvement of our complete model over the best baseline is significant at the confidence level of 0.95.



(a) Precision@10 (b) Recall@10

Fig. 12. Performance comparison for overall recommendation with *the solved problems* as ground truth on three datasets. The improvement of our complete model over the best baseline is significant at the confidence level of 0.95.

of our model is that it can utilize auxiliary information to enhance the mode learning. In recommendation, we only utilize the features extracted from the preceding attempts.

- By examining the variants of our model, it is easy to see that modeling interests is important for recommendation. Similar to the finding in Figure 11 and 12, the improvement of the complete model over the variant *our model*$_{\neg exp}$ in terms of the solved-problem recommendation is more significant than the attempted-problem recommendation.

Summary. By combing the results in Figure 11, 12 and Table 9, we can find our proposed model can make more accurate problem recommendation by matching both the interest and expertise of a user.

6 RELATED WORK

In this section, we review the related work in four aspects. The first two aspects are mainly related to our tasks, while the last two aspects are mainly related to our models. We also discuss the major differences of our proposed approaches in comparison with the previous work.

Table 9. Performance comparison for next-basket recommendation using hit@10 on three datasets. "**" indicates the improvement over the best baseline is significant at the confidence level of 0.99.

Methods	Attempted problems			Solved problems		
	POJ	Timus	HDU	POJ	Timus	HDU
TOP	0.139	0.194	0.122	0.136	0.176	0.171
MC	0.145	0.336	0.183	0.106	0.354	0.276
FPMC	0.274	0.322	0.297	0.291	0.336	0.302
HRM	0.287	0.339	0.312	0.309	0.363	0.351
our model$_{\neg exp}$	0.346	0.396	0.355	0.331	0.379	0.369
our model$_{\neg int}$	0.027	0.104	0.085	0.078	0.091	0.087
our model	**0.355****	**0.402****	**0.388****	**0.389****	**0.412****	**0.407****

6.1 Cognitively Diagnostic Assessments and Knowledge Tracing

Our work is closely related to the Cognitively Diagnostic Assessments (CDA) originated from educational psychology [19, 38]. CDA aims to diagnose students' strengths and weaknesses and to provide specific information in the form of skill mastery profiles using Cognitive Diagnosis Models. Traditional models often employed item response theory (IRT) or classical test theory (CTT), which was developed by modeling the probability that a learner was able to correctly solve a problem [20]. For CDA, a key task is to measure the specific ability level of a learner on multiple skills or knowledge components. A widely used mathematical formulation is the well known Q-matrix [3, 58], which is typically a binary matrix establishing the relationship between responses and attributes (i.e., skills) by indicating the required attributes for each question. To construct a Q-matrix, experts in a particular domain usually need to specify the correspondence between attributes and key knowledge or skills for students to acquire [16]. The manual construction of the Q-matrix in the early stage is time-consuming. It is therefore essential to construct the Q-matrix automatically. During the past decade, the problem of how to map test questions into latent skills based on students' test responses has become a hot research topic in Educational Data Mining (EDM) [17, 26]. Typical approaches include the well-known DINA model based Q-matrix learning approach [14, 15], which adopts the EM algorithm to solve the Q-matrix. By factorizing the user-question response data matrix, Matrix Factorization (MF) has been applied to automatically learn the Q-matrix from the students' item response matrix [17, 18]. More recently, fuzzy cognitive diagnosis framework has been proposed to model the skill proficiency of examinees [61].

Knowledge tracing is a common task in EDM, which aims to model student knowledge over time for the predictions of students' performance in near future [11]. A popular knowledge tracing approach is Bayesian Knowledge Tracing (BKT) [64]. BKT characterizes a learner's latent knowledge state as a set of binary variables, corresponding to the learners' understandings on different concepts. Furthermore, such knowledge states are traced in a sequential manner based on Markov models. Many extensions have been proposed to improve BKT, including contextualization of guessing and slipping estimates [12] and estimating prior knowledge for individual learners [39]. More recently, various approaches have been developed by applying new data models to knowledge tracing, such as Markov decision processes [45], sparse factor analysis [31], and recurrent neural networks [42]. These sequence models typically assume that the current state depends on historical information but did not explore well users' sequential behavior characteristics.

6.2 User Learning Behaviors/Characteristics

Our work is related to the traditional studies on behavioral characteristics or patterns in the learning process. Bransford et al. [6] presented a thorough and comprehensive summarization and review of human being learning in various aspects, including learners and learning, teachers and teaching, and learning environments. One interesting topic is to study how the arrangement of practice sets affects the learning of students [51]. Two fundamental ways of arrangement are blocked and mixed view. Mixed view refers to distributing practice questions on the same topic across many practice sets, while blocked view refers to organizing practice questions on the same topic in a single practice set. With the mixed view, it has been found that spacing provides review and improves long-term retention, and mixing improves students' ability to pair a problem with the appropriate concept and procedure [52].

Our work is also related to self-regulated learning, which refers to learning that is guided by metacognition and motivation to learn [56]. As an important research subject, metacognition describes the processes involved when learners plan, monitor, evaluate and make changes to their own learning behaviors. Based on the concept of metacognition, "self-regulated" describes a process of taking control of and evaluating one's own learning and behavior [66].

Furthermore, our work is related to the mining of programming trace data, including the development of online judges [35, 40, 50], hint generation in problem solving [43], and modeling how students learn to program [44].

6.3 Expertise Learning using Comparison-based Competitions

Measuring the question difficulty and student ability is related to the work of learning and mining expertise in both general and domain-specific settings. In games and matches, the competition-based ranking methods have also been widely studied [13, 21], where the aim is to evaluate the skill level of each involved player. These studies typically only use a scalar value as the measure of the skill rating of an individual player. By extending the two-player model, TrueSkill [28] is able to characterize a multi-player setting by incorporating a dynamic variance.

In sociology, it is a common sense that competition is usually correlated with expertise [62]. Following this, many studies try to model the expertise level of a user using a competition-based ranking approach, e.g., community question and answering platforms [33] and generalized crowdsourcing systems [10]. More recently, Chen et al. [8, 9] have proposed to use low-dimensional latent representations for modeling match-ups and preferences. By modeling comparison-based preference, we can essentially perform any ranking task. For example, in information retrieval (IR), learning to rank aims to learn the ranking for a list of candidate items with manually selected features [34].

Another related topic is to mine and measure expertise using social data. Since the direct performance data is infeasible to obtain in many scenarios, much research tried to leverage social data to estimate the expertise level of users, including enterprise-specific social data [25], community question and answering data [63], and microblogging data [65].

6.4 Topic Modeling and User Interest Profiling

Topic modeling is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. The pioneering models include Probabilistic Latent Semantic Analysis (PLSA) [29] and LDA [5] which mainly characterize a set of topics and document-specific distributions over these topics. The topics are modeled by multinomial distributions over the terms in a vocabulary, and the top ranked terms in a topic are supposed to provide a word-level semantic explanations for the underlying topic. The standard topic models (e.g., LDA) make the exchangeability

assumption for the word tokens in a document. Subsequently, the assumption was relaxed in many ways, such as the modeling of the topic correlation [4] and hierarchical topic structure [59]. The most relevant study is the incorporation of topic dependence between consecutive word tokens. A commonly adopted strategy is to assume the topic label of the current word depends on that of the previous word, so called *hidden Markov topic model* [23, 24].

Furthermore, topic models have been applied to learn user interests in many tasks. A representative work is the Author-Topic Model [55], in which an author has an interest distribution over topics in text generation. More recently, topic models have been adapted to model users interest profiles in various applications, including trajectory mining [32], sentiment analysis [46], and recommender system [36].

Differences with Previous Work. Our work focuses on improving OJ system by mining users' learning trace data. In particular, we select the programming OJ systems [50] as the testbed. Mining users' learning traces from online learning platforms, such as traditional intelligent tutoring systems [2, 30] and MOOC [1], has received much attention in the past. Most of the existing studies have been conducted on private and/or well-organized learning or tutoring systems, in which the students are well monitored and instructed in the process of learning. We also aim to understand how students learn and behave on online learning platforms [53], but with a particular focus on OJ systems, which represent public and self-regulated learning systems. Since there is a lack of supervisions and feedbacks from teachers, our task setting is more challenging than that in existing studies. To the best of our knowledge, our work is the first that focuses on understanding users' learning behaviors by mining users' learning traces on OJ systems.

7 CONCLUSION

In this work, we studied how to automatically mine topic and difficulty information from users' learning traces on OJ systems. Our analysis on users' learning traces leads to an important finding: there are two major learning modes on OJ systems, where users either choose questions sequentially from the same volume regardless of their topics or identify questions from multiple volumes but all about the same topic for practice. Based on this observation, we proposed a novel two-mode Markov topic model, which can jointly characterize the two learning modes, for topic detection of online problems. With the learned topic information, we developed a topic-aware competition-based expertise model. Extensive experiments on three large online judge datasets have demonstrated the effectiveness of our approach in three different tasks, including skill topic extraction, expertise competition prediction and problem recommendation. We believe our work will be of great value to improve online education services.

In our current work, we mainly consider the sequential characteristics in users' learning behaviors. On programming OJ systems, topics and difficulty levels of problems remain relatively stable, but the topical interests and expertise levels of users are likely to evolve over time. Recently, recurrent neural networks have been shown to be effective as a general approach to modeling temporal dynamics and dependency in sequence data. It is thus possible to model users' interest and expertise evolvement using sequential neural network models in future work. Also, we did not utilize problem descriptions here. We will study how to extract useful topic information from problem descriptions by jointly modeling text content and user behavior. As the future work, we will also consider evaluating our proposed approaches on other types of online learning or question-answering systems, such as MOOC and Quora.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their valuable and constructive comments.

REFERENCES

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

[2] John R Anderson, C Franklin Boyle, and Brian J Reiser. 1985. Intelligent tutoring systems. *Science(Washington)* 228, 4698 (1985), 456–462.

[3] Tiffany Barnes, Donald L. Bitzer, and Mladen A. Vouk. 2005. Experimental Analysis of the Q-Matrix Method in Knowledge Discovery. In *Foundations of Intelligent Systems, 15th International Symposium, ISMIS 2005, Saratoga Springs, NY, USA, May 25-28, 2005, Proceedings*. 603–611.

[4] David M Blei and John D Lafferty. 2007. A correlated topic model of science. *The Annals of Applied Statistics* (2007), 17–35.

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

[6] John D. Bransford, Ann L. Brown, and Rodney R. Cocking. 2000. *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*. National Academy Press., Washington, DC.

[7] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*. ACM, 1–10.

[8] Shuo Chen and Thorsten Joachims. 2016. Modeling intransitivity in matchup and comparison data. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 227–236.

[9] Shuo Chen and Thorsten Joachims. 2016. Predicting matchups and preferences in context. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 775–784.

[10] Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 193–202.

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

[12] Ryan SJ d Baker, Albert T Corbett, and Vincent Aleven. 2008. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International Conference on Intelligent Tutoring Systems*. Springer, 406–415.

[13] Pierre Dangauthier, Ralf Herbrich, Tom Minka, and Thore Graepel. 2008. Trueskill through time: Revisiting the history of chess. In *Advances in Neural Information Processing Systems*. 337–344.

[14] Jimmy de la Torre. 2009. DINA model and parameter estimation: A didactic. *Journal of Educational and Behavioral Statistics* 34, 1 (2009), 115–130.

[15] Jimmy de la Torre and Jeffrey A. Douglas. 2004. Higher-order latent trait models for cognitive diagnosis. *Psychometrika* 69, 3 (2004), 333–353.

[16] Michel Desmarais, Behzad Beheshti, and Peng Xu. 2014. The refinement of a Q-matrix: Assessing methods to validate tasks to skills mapping. In *Proceedings of the 7th International Conference on Educational Data Mining, EDM 2014, London, UK, July 4-7, 2014*. 208–311.

[17] Michel C Desmarais. 2012. Mapping question items to skills with non-negative matrix factorization. *ACM SIGKDD Explorations Newsletter* 13, 2 (2012), 30–36.

[18] Michel C Desmarais and Rhouma Naceur. 2013. A matrix factorization method for mapping items to skills and for enhancing expert-based q-matrices. In *International Conference on Artificial Intelligence in Education*. Springer, 441–450.

[19] Louis V DiBello, William F Stout, and Louis A Roussos. 1995. Unified cognitive/psychometric diagnostic assessment likelihood-based classification techniques. *Cognitively diagnostic assessment* (1995), 361–389.

[20] S. E. Embretson and S. P. Reise. 2000. *Item response theory for psychologists*. Lawrence Erlbaum, Mahwah.

[21] Mark E Glickman. 1995. A comprehensive guide to chess ratings. *American Chess Journal* 3 (1995), 59–102.

[22] Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences* 101, suppl 1 (2004), 5228–5235.

[23] Thomas L Griffiths, Mark Steyvers, David M Blei, and Joshua B Tenenbaum. 2004. Integrating Topics and Syntax.. In *NIPS*, Vol. 4. 537–544.

[24] Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden Topic Markov Models.. In *AISTATS*, Vol. 2. 163–170.

[25] Ido Guy, Uri Avraham, David Carmel, Sigalit Ur, Michal Jacovi, and Inbal Ronen. 2013. Mining expertise and interests from social media. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 515–526.

[26] Edward H Haertel. 1989. Using restricted latent class models to map the skill structure of achievement items. *Journal of Educational Measurement* 26, 4 (1989), 301–321.

[27] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[28] Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. TrueSkillâĎć: a Bayesian skill rating system. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*. MIT Press, 569–576.

[29] Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 50–57.

[30] James A Kulik and JD Fletcher. 2016. Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of Educational Research* 86, 1 (2016), 42–78.

[31] Andrew S Lan, Christoph Studer, and Richard G Baraniuk. 2014. 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*. ACM, 452–461.

[32] Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. 2013. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1043–1051.

[33] Jing Liu, Young-In Song, and Chin-Yew Lin. 2011. Competition-based user expertise score estimation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 425–434.

[34] Tie-Yan Liu and others. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[35] Shahriar Manzoor. 2006. Analyzing Programming Contest Statistics. *Perspectives on Computer Science Competitions for (High School) Students* (2006), 48.

[36] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 165–172.

[37] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[38] Paul D Nichols, Susan F Chipman, and Robert L Brennan. 2012. *Cognitively diagnostic assessment*. Routledge.

[39] Zachary A Pardos and Neil T Heffernan. 2010. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *International Conference on User Modeling, Adaptation, and Personalization*. Springer, 255–266.

[40] Jordi Petit, Omer Giménez, and Salvador Roura. 2012. Jutge. org: an educational programming judge. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, 445–450.

[41] Xuan-Hieu Phan and Cam-Tu Nguyen. 2007. GibbsLDA++: AC/C++ implementation of latent Dirichlet allocation (LDA). *Tech. rep.* (2007).

[42] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*. 505–513.

[43] Chris Piech, Mehran Sahami, Jonathan Huang, and Leonidas Guibas. 2015. Autonomously generating hints by inferring problem solving policies. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*. ACM, 195–204.

[44] Chris Piech, Mehran Sahami, Daphne Koller, Steve Cooper, and Paulo Blikstein. 2012. Modeling how students learn to program. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, 153–160.

[45] Anna Rafferty, Emma Brunskill, Thomas Griffiths, and Patrick Shafto. 2011. Faster teaching by POMDP planning. In *Artificial intelligence in education*. Springer, 280–287.

[46] Md Mustafizur Rahman and Hongning Wang. 2016. Hidden Topic Sentiment Model. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 155–165.

[47] PV Rao and Lawrence L Kupper. 1967. Ties in paired-comparison experiments: A generalization of the Bradley-Terry model. *J. Amer. Statist. Assoc.* 62, 317 (1967), 194–204.

[48] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[49] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 811–820.

[50] Miguel A Revilla, Shahriar Manzoor, and Rujia Liu. 2008. Competitive learning in informatics: The UVa online judge experience. *Olympiads in Informatics* 2 (2008), 131–148.

[51] Doug Rohrer. 2009. The Effects of Spacing and Mixing Practice Problems. *Journal for Research in Mathematics Education* 40, 1 (2009), 4–17. http://www.jstor.org/stable/40539318

[52] Doug Rohrer and Kelli Taylor. 2006. The effects of overlearning and distributed practise on the retention of mathematics knowledge. *Applied Cognitive Psychology* 20, 9 (2006), 1209–1224. DOI:https://doi.org/10.1002/acp.1266

[53] Cristobal Romero and Sebastian Ventura. 2007. Educational data mining: A survey from 1995 to 2005. *Expert systems with applications* 33, 1 (2007), 135–146.
[54] Alexander Joseph Romiszowski. 2016. *Designing instructional systems: Decision making in course planning and curriculum design*. Routledge.
[55] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 487–494.
[56] Steven V Shannon. 2008. Using metacognitive strategies and learning styles to create self-directed learners. *Institute for Learning Styles Journal* 1, 1 (2008), 14–28.
[57] Robert E Slavin and Nicola Davis. 2006. Educational psychology: Theory and practice. (2006).
[58] Yuan Sun, Shiwei Ye, Shunya Inoue, and Yi Sun. 2014. Alternating Recursive Method for Q-matrix Learning. In *Proceedings of the 7th International Conference on Educational Data Mining, EDM 2014, London, UK, July 4-7, 2014*. 14–20.
[59] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2004. Sharing Clusters among Related Groups: Hierarchical Dirichlet Processes.. In *NIPS*. 1385–1392.
[60] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 403–412.
[61] Run-ze Wu, Qi Liu, Yuping Liu, Enhong Chen, Yu Su, Zhigang Chen, and Guoping Hu. 2015. Cognitive Modelling for Predicting Examinee Performance. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. 1017–1024.
[62] Jiang Yang, Lada A Adamic, and Mark S Ackerman. 2008. Competing to Share Expertise: The Taskcn Knowledge Sharing Community.. In *ICWSM*.
[63] Liu Yang, Minghui Qiu, Swapna Gottipati, Feida Zhu, Jing Jiang, Huiping Sun, and Zhong Chen. 2013. Cqarank: jointly model topics and expertise in community question answering. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 99–108.
[64] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. 2013. Individualized bayesian knowledge tracing models. In *International Conference on Artificial Intelligence in Education*. Springer, 171–180.
[65] Wayne Xin Zhao, Jing Liu, Yulan He, Chin-Yew Lin, and Ji-Rong Wen. 2016. A computational approach to measuring the correlation between expertise and social media influence for celebrities on microblogs. *World Wide Web* 19, 5 (2016), 865–886.
[66] Barry J Zimmerman, Dale H Schunk, Anita Woolfolk Hoy, and Pamela J Gaskill. 2003. Self-regulated learning. *Psyccritiques* 48, 1 (2003), 16–18.

Received May 2017; revised August 2017; accepted November 2017