Topic Transitions in MOOCs: An Analysis Study

Fareedah ALSaad*
University of Illinois at Urbana-Champaign
alsaad2@illinois.edu

Thomas Reichel
University of Illinois at Urbana-Champaign
reichel3@illinois.edu

Yuchen Zeng
University of Illinois at Urbana-Champaign
yuchenz8@illinois.edu

Abdussalam Alawini
University of Illinois at Urbana-Champaign
alawini@illinois.edu

ABSTRACT

With the emergence of MOOCs, it becomes crucial to automate the process of a course design to accommodate the diverse learning demands of students. Modeling the relationships among educational topics is a fundamental first step for automating curriculum planning and course design. In this paper, we introduce *Topic Transition Map* (TTM), a general structure that models the content of MOOCs at the topic level. TTMs capture the various ways instructors organize topics in their courses by modeling the transitions between topics. We investigate and analyze four different methods that can be exploited to learn the Topic Transition Map: 1) Pairwise Constrained K-Means, 2) Mixture of Unigram Language Model, 3) Hidden Markov Mixture Model, and 4) Structural Topic Model. To evaluated the effectiveness of these methods, we qualitatively compare the topic transition maps generated by each model and investigate how the Topic Transition Map can be used in three sequencing tasks: 1) determining the correct sequence, 2) predicting the next lecture, and 3) predicting the sequence of lectures. Our evaluation revealed that PCK-Means has the highest performance in the first task, HMMULM outperforms other methods in task 2, while there is no winning in task 3.

Keywords

Topic Transition Map, Topic Transition, Word Distribution, Mixture Model, Hidden Markov Model, Clusters, Sequencing Tasks.

1. INTRODUCTION

For many decades, the process of creating courses has been a manual task that needs to be carefully managed by instructors and experts. However, with the recent advances in technologies and the emergence of Massive Open Online Courses (MOOCs), it becomes critical to automate the process of course design to accommodate the heterogeneity of online students and their diverse needs. According to [32], learning on demand is considered one factor that causes the high dropout rate in MOOCs. Learners have different learning demands depending on their motivations and goals. For instance, learners may seek knowledge about an interdisciplinary domain and hence need to learn modules from courses in several areas. This problem requires adopting a model in which MOOCs are used as modularized resources, rather than a set of pre-designed static courses. A crucial first step toward developing such a model is the automation of course plan design by sequencing lectures among different courses.

The main principle in designing the curriculum of any course is to organize course content according to some relations between topics. For instance, to help students to learn the materials, instructors carefully organize lectures as a sequence, based on the difficulty levels of topics [10, 27, 1] as well as the dependency relations between topics [11, 21, 23, 1]. The fundamental sequential structure of a course design is to place topics that are easy or prerequisite in earlier lectures while more advanced and dependent topics are taught in later lectures [1]. Consequently, modeling the relatedness among educational topics is a very crucial first step for automating curriculum planning and course design.

Modeling the content structure of MOOCs has recently attracted much research. Most of the current research has focused on modeling the prerequisite relationships between courses [29, 15], between lectures or segments of lectures [6, 7], or between concepts discussed within or across courses [3, 14, 17, 29, 15]. Using concepts to model MOOCs' content can be easily generalized to capture the relations in the concept space. However, because concepts are represented as keywords or phrases, it is hard to capture the different levels of granularity between lectures and courses. In addition, modeling prerequisite relationships between concepts cannot capture the various learning paths accommodated by different courses.

In this paper, we introduce the **Topic Transition Map**, a general structure that models the educational materials at the topic level. We model a course as a set of topics, and each topic is a set of concepts. Modeling content at the topic level is a more natural way to design custom course plans. We can think of a course as a path in the generalized Topic Transition Map. Thus, designing a new course becomes a task of identifying a path in the Topic Transition Map. Additionally, we investigate four methods that can be leveraged to construct the Topic Transition Map: *Pairwise Constrained K-Means* (PCK-Means) [2], *Mixture of Unigram Language Model* (MULM), *Hidden Markov Mixture Model* (HMMULM), and *Structural Topic*

*King AbdulAziz University, Jeddah, Saudi Arabia.

Model (strTM) [24]. We analyze and compare the Topic Transition Maps learned by these methods by studying how to exploit the Topic Transition Map in three sequencing tasks: 1) determining the correct sequence, 2) predicting the next lecture, and 3) predicting the sequence of lectures. To the best of our knowledge, we are the first work to introduce and investigate the use of Topic Transition Maps in modeling MOOCs content and sequencing lectures.

To evaluate the effectiveness of all methods, we use real MOOCs from three different domains: Python, Structural Query Language, and Machine Learning Clustering algorithms. Our evaluation revealed that while the PCK-Means has the highest performance on the task of finding the best sequence from a list of possible sequences, the HMMULM achieves the best performance on the task that predicts the next lecture in the sequence. Additionally, all methods perform similarly in the task of predicting the whole sequence with MULM has the lowest performance as it sometimes cannot predict the whole sequence. In addition to comparing various models in sequencing tasks, we visualize the Topic Transition Maps generated by different methods to qualitatively compare the resulted topic transition maps. We found that PCK-Means has extracted more meaningful topics with the best word distributions that clearly explain each topic.

The rest of the paper is organized as follows. In section 2, we present some of the related work. Section 3, defines the topics and topic transitions and states some applications of the topic transition maps. In section 4, we formally define our problem before describing the four different methods we exploit to construct the topic transition maps in section 5. Section 6 elaborates on our approach for the evaluation and the analysis of various models. Finally, we conclude our work in section 7.

2. RELATED WORK

Most of the work that models the content of MOOCs has focused on capturing the prerequisite relationships using different levels of granularity such as courses [29, 15], lectures or segments of lectures [6, 7], or concepts discussed within or across courses [3, 14, 17, 29, 15]. Modeling the relations between courses, lectures, or seqments of lectures is restricted to these units and cannot be generalized. While modeling dependency relations between concepts is considered a general structure that captures the required concepts before learning any concept, prerequisite relations cannot model the various learning paths accommodated by different courses. ALSaad and Alawini [2] have addressed this problem by proposing the precedence graph that captures the similarities and variations of learning paths among different courses. We build on their work and introduce the **Topic Transition Map** that maps each lecture to a topic and leverages the sequences of lectures among courses to capture the topic transitions pattern and hence the likelihood of such a transition. The main difference between the Topic Transition Map and the precedence graph is that Topic Transition Map models self transitions between a topic and itself and also captures how likely each topic to be the first topic in courses. While ALSaad and Alawini [2] have investigated the use of PCK-Means in modeling the precedence graph, in this paper, we explore three more methods in addition to PCK-Mean, namely MULM, HMMULM, and strTM, for modeling Topic Transition Maps. We also examine the impact of the learned topic transition maps on three different sequencing tasks. We believe that we are the first work that examines the use of topic transitions modeled from existing MOOCs to learn how to sequence new courses.

Some research has investigated the use of prerequisite relations between concepts to construct and sequence learning units [1, 16]. Both studies [1, 16] have developed supervised approaches based on feature engineering that extracted features from some external knowledge such as Wikipedia [1] and DBpedia [16] to infer the prerequisite relations between concepts. Our work is different as instead of modeling the prerequisite relations between concepts using supervised approaches, we model the Topic Transition Map or the various paths between topics using unsupervised methods, where a topic is a set of concepts. In addition, our methods rely only on the content of MOOCs without using any external knowledge. While Agrawal et al. [1] used the concept dependency graph to organize concepts to construct learning units and then sequence the learning units, we use lectures from existing MOOCs and investigate the impact of the learned transitions between topics to sequence lectures.

The most relevant research to our study is the work by Shen et al. [22]. Shen et al. [22] have proposed a method for linking similar courses to construct a map of lectures connected by two types of relations: similar and prerequisite. The constructed map only captures the similarity and prerequisite relations between certain units (lectures) and is not generalized to other lectures and thus cannot be used to predict the sequence of new lectures. In this paper, we map lectures to topics and construct the Topic Transition Map that depicts the precedence relations between topics and hence not tied with any specific units. Having a generalized Topic Transition Map can help in finding the sequence of lectures or predict the next lecture in the sequence as we discuss in section 6.2.

Another related line of research is the work on structural topic modeling by the Natural Language Processing, NLP, Community. In NLP, topic transitions have been used to model latent topical structures inside documents by assuming each sentence is generated from a topic where topics satisfy the first order Markov property [12, 25]. While Gruber et al. [12] only modeled the transition between topics as a binary relation (either remain on the current topic or shift to a new topic with a certain probability), Wang et al. [25] have developed a Structural Topic Model called strTM to explicitly model the topic transitions as probabilities that capture how likely one transits from a topic to another. Modeling transitions have been used in many applications related to NLP such as sentence ordering [25], topic segmentation [9], and multi-documents summarization [28]. In this paper, we investigate the use of topic transitions on modeling the topical structures in MOOCs by assuming a lectures is generated by one topic and use the sequences of lectures to learn the transitions between topics. We also explore the impact of using the Topic Transition Map to sequence lectures in three different sequencing tasks.

3. TOPIC TRANSITIONS

Before defining the topic transitions, it is important to briefly explain our representation of topics used in this paper. Similar to the definition of topics in the literature of the topic modeling research [5, 13], we define a topic as a distribution of concepts where concepts with higher probabilities tend to explain or characterize the topic. Concepts can be represented as words or phrases of words [3, 18, 26]. Each lecture is a composition of concepts and hence can be mapped to some topics. Depending on the length of lectures, lectures can cover one or more topics. Longer lectures usually cover more topics than shorter lectures. For example, traditional university lectures tend to be more elaborated and have longer duration than MOOCs lectures, which are usually concise and short in length. Therefore, the number of topics per lecture discussed in MOOCs is less than that of traditional university lectures. In this paper, since our work focus on learning the topic transitions from MOOCs, we assume

that each lecture is mapped only to one topic. This assumption is reasonable as lectures in MOOCs are concise and short in length. Having this assumption is also very useful as it helps in leveraging the sequences of lectures to learn the relations between topics.

A topic transition captures the precedence relations between topics. In other words, it means how likely instructors move or transit from one topic to another in the course delivery. It models the various ways of how instructors dynamically assemble concepts from the concepts space in order to construct the study plan of their courses. For instance, some instructors decide to start their *Python* course by explaining the topics: data types, conditional statements, loops, and then reading and writing from files. Other instructors may choose a different order, such as conditional statements, loops, string, and then lists. By leveraging the sequences of lectures from multiple courses, we can infer the latent topics of each lecture and hence model the common transition patters shared by multiple courses as well as the variations of different transitions or paths. To determine the strength or how common that transition is, each transition is attached with a score or a probability. For example, given the topics in Computer Science Programming: "Conditional Statements", "Loops", and "Arrays". It is more likely that instructors will explain the topic "Conditional Statements" immediately before the topic "Loops" and thus the topic transition score between them would be higher than the transition score between the topic "Conditional Statements" and the topic "Arrays".

Learning the topic transitions can be the initial block to facilitate several useful applications that can support modern learning. For instance, we can use the Topic Transition Map to extract the most common paths of topics in the field or explain the topic space in the current MOOC offerings. Learners can use transition maps to get more insights about the structure of topics in MOOC offerings. On the other hand, instructors can use these maps to improve their course offerings by examining the topic structure of related courses.

One important application of the Topic Transition Map is to support automatic curriculum planning and course design. Since courses consist of topics, learning the relations between topics would be the initial step to understand how likely instructors transit from one topic to another. We can think of a course as a path in the generalized Topic Transition Map. Thus, designing a new course becomes a task of identifying a path in the Topic Transition Map. In this paper, we analyze how can we use the learned topic transitions to sequence new courses.

4. PROBLEM FORMULATION

In this section, we formally formulate our problem. Given a set of courses $C = \{X_1, X_2, X_3, \ldots, X_N\}$ from a particular domain, where N is the total number of courses. We assume that courses in C are similar and hence have some content overlaps between them and also have the same difficulty level (e.g. Beginner, Medium, or Advance). A course X_i is represented as an ordered list of lectures $X_i = [x_{i1}, x_{i2}, \ldots, x_{i|X_i|}]$, where $|X_i|$ is the total number of lectures in the course X_i. Each lecture is a composition of concepts represented in some narrative way. In this paper, we assume a concept as a single word and hence lectures are represented using a bag-of-word representation.

Given the number of topics M, our goal is to map each lecture to a topic and leverage the sequences of lectures to learn topic transitions and construct the Topic Transition Map. The Topic Transition Map is represented as a matrix $\mathbf{A}$, where $\mathbf{A} \in \mathbb{R}^{M \times M}$. Each entry a_{ij} of the matrix $\mathbf{A}$ represents the likelihood of the transition from topic i to topic j. It reflect how common the precedence relation from topic i to j in the dataset courses. In addition to the Topic Transition Map, we also aim to learn the probability of each topic being an initial topic in courses. We denote the initial probability of each topic as a vector π, where $\pi \in R^M$. Along with the Topic Transition Map and the initial probability of each topic, it is important to model the word distribution of each topic, which represented as a matrix $\mathbf{B} \in \mathbb{R}^{M \times V}$, where V is the vocabulary size.

5. MODELING TOPIC TRANSITIONS

In this section, we explain the four different models we exploit to capture topics and Topic Transition Maps.

5.1 Pairwise Constrained K-Means

PCK-Means clustering algorithm [4] is a variation of the standard K-Means algorithm. To cluster instances, PCK-Means incorporates distance between points as well as pairwise constraints to guide the clustering process. Since the purpose of clustering is to capture topic transition patterns across courses, using PCK-Means helps to restrict the clustering process to cluster lectures across courses instead of within courses [2]. To guide the clustering, PCK-Means uses two types of constraints: **Must-Link** and **Cannot-Link**. Must-Link constraint determines lecture pairs that need to be clustered together, while Cannot-Link constraint specifies pairs that should not be grouped into the same cluster. To find the clusters, PCK-Means uses an objective function that minimizes both: 1) the distance between points (lectures) and the cluster centroid, and 2) the penalty costs of violating the constraints. For more information about PCK-Menas, please refer to [4].

Similar to ALSaad and Alawini [2], we use PCK-Means to build the Topic Transition Map $\mathbf{A}$. We first construct the list of Must-Link and Cannot-Link constraints to clusters lectures based on their content similarity into clusters. We assume that each cluster forms a topic and hence we need to learn the word distributions of each topic along with topic transitions. We link clusters by using the precedence relations between adjacent lectures and capture the strength of the transition by accumulating the frequency of transitions. To find the word distribution of each cluster or topic in the matrix $\mathbf{B}$, we accumulate the vector representations of each lecture that belongs to the same cluster. For more information, please see [2].

In order to estimate the initial probability π for each topic, we simply count the number of times of each topic being the first topic in the set of courses C. Then we do normalization to find the probability.

5.2 Mixture of Unigram Language Models

To capture topics, we use a mixture model of M unigram language models (MULM) with a bag-of-words representation. The mixture model is a generative probabilistic model that has been used for documents clustering. Thus, it will help in clustering lectures based on their topics, where each lecture belongs only to one cluster or one topic. In the mixture model, to generate a document, first one needs to choose the topic of the document according to the probability $P(\theta_i)$, where M is the number of topics, and then generate all the words in the document using the probability $P(w|\theta_i)$. According to the model, the likelihoods of a document x and the corpus C are calculated as follows:

$$P(x|\lambda) = \sum_{i=1}^{M} P(\theta_i) \prod_{w \in V} P(w|\theta_i)^{c(w,x)} \quad (1)$$

$$P(C|\lambda) = \prod_{j=1}^{N} P(x_j|\lambda) \tag{2}$$

To estimate the model parameters $\lambda = (\{\theta_i\}, P(\theta_i))$, where θ_i is the word distribution of topic i from the matrix $\mathbf{B}$, we use Expectation-Maximization algorithm [8] to find the parameters that maximize the likelihood of the data:

$$\bar{\lambda} = \arg\max_{\lambda} P(C|\lambda) \tag{3}$$

After learning the parameters, we map each lecture to a cluster or topic by maximizing the following equation:

$$c = \arg\max_{z_i} P(x|z_i) \tag{4}$$

Using the mixture model can help in clustering lectures according to their topics, however; it will not capture the transition patterns between topics or the initial probability of each topic. Therefore, similar to the PCK-Means method, we leverage the sequences of lectures to calculate the score of topic transitions and construct the Topic Transition Map $\mathbf{A}$. Likewise, we count the number of times courses start with each topic and normalize the results to model the initial probability π.

5.3 Hidden Markov Mixture Models

Instead of separately clustering lectures and then learning the transitions between them, using a Hidden Markov Model would allow us to jointly learn the word distributions of each topic ($\mathbf{B}$), the transition probabilities between topics ($\mathbf{A}$) as well as the initial probability of each topic (π).

The Hidden Markov Model, HMM, is a probabilistic graphical model that describes the process of generating a sequence of observable events according to some hidden factors [20]. It simulates how the real world sequence data is generated from hidden states. Particularly, it consists of two stochastic processes: 1) invisible process, and 2) visible process [30]. In HMM, invisible process consists of hidden states whereas visible process is observed sequence of symbols that are drawn from the probability distributions of the hidden states. Figure 1 demonstrates the HMM model. As you can see from Figure 1, each observable event in the sequence are generated from a hidden state and observations are conditionally independent given the hidden state. You can also notice that the hidden states form a Markov chain where each hidden state depends only on the previous state such as Z_{t+1} depends on Z_t.

To control the process of generating the observed sequences from hidden states, HMM has three parameters: π, $\mathbf{A}$, and $\mathbf{B}$. The first parameter, $\pi = \pi_1, \pi_2, \ldots, \pi_M$, is the initial probability distribution of each hidden state. The parameter π determines the probability of the Markov chain to start at each state and hence controls which state can be chosen as an initial state for the observed sequence. The second parameter, $\mathbf{A} \in \mathbb{R}^{M \times M}$, is the transition probability matrix that specifies how likely the model can transit from one state to another, denoted by $P(Z_{t+1}|Z_t)$ in Figure 1. The third parameter, $\mathbf{B} \in \mathbb{R}^{M \times V}$, is the emission probability matrix, where V is the total number of distinct symbols. It determines the likelihood of each state to produce each symbol, denoted by $P(X_{t+1}|Z_{t+1})$ in Figure 1. For example, to generate a sentence, a sequence of words would be drawn from the HMM model according to the three parameters π, $\mathbf{A}$, and $\mathbf{B}$.

In MOOCs, we only observe courses, where courses are sequence of lectures, while the topics of lectures and the transition between them are invisible or latent. Therefore, HMM would be a great model to simulate the generation process of courses and hence infer the latent states that contribute in the evolution of these lectures. In HMM, each hidden state generates only one symbol or word (see Figure 1). As our goal is to capture topic transitions using sequences of lectures as observed data, we map each lecture to a topic and assume each hidden state generates a lecture instead of a word. Our revised HMM assumes that each hidden state produces one lecture where each lecture is a bag-of-words. We ignore the sequence of words in lectures since the order of the words would not contribute to capturing the topic of each lecture. Figure 2 depicts the HMMULM utilized to capture the content of MOOCs.

In order to capture both the lectures' topics and the transitions between them, we combine the mixture model (MULM) with HMM, and we call the new model Hidden Markov Mixture of Unigram Language Model (HMMULM). To do that, we assume the Markovian assumption between topics where in the generation process, the choice of the next topic depends only on the current topic. Even though, the choice of the topic in the course delivery depends on the previous topics discussed so far, this simplified assumption makes sense due to the locality of reference property [1] of course design. Based on this property, when an instructor designs a course, a dependent lecture should appear as soon as possible after the prerequisite lecture to reduce students comprehension burden. Therefore, assuming the dependency between adjacent lectures not only simplifies the model but also aids in capturing the transitions between highly related topics. By combining the HMM with mixture model the likelihood of generating a course is as follow:

$$\begin{aligned}
P(X|\lambda) &= \sum_{all\ Z} P(Z|\lambda) P(X|Z,\lambda) \\
&= \sum_{all\ Z} P(z_1) P(x_1|z_1) \prod_{t=2}^{T} P(z_t|z_{t-1}) P(x_t|z_t) \\
&= \sum_{all\ Z} P(z_1) \prod_{w \in V} P(w|z_1)^{c(w,x_1)} \\
&\prod_{t=2}^{T} P(z_t|z_{t-1}) \prod_{w \in V} P(w|z_t)^{c(w,x_t)} \\
&= \sum_{i=1}^{M} \sum_{j=1}^{M} \pi(z_1 = s_i) \prod_{w \in V} B(z_1 = s_i, w)^{c(w,x_1)} \\
&\prod_{t=2}^{T} A(z_{t-1} = s_i, z_t = s_j) \prod_{w \in V} B(z_t = s_j, w)^{c(w,x_t)}
\end{aligned} \tag{5}$$

To estimate the HMMULM parameters $\lambda = (\pi, A, B)$, we use a modified version of Baum-Welch algorithm in order to model the observation sequences as a multidimensional categorical events. Following the work [19], we derived the equations of E-step and M-step to train the model and infer the transition probability between topics. In the E-step, we use the equations:

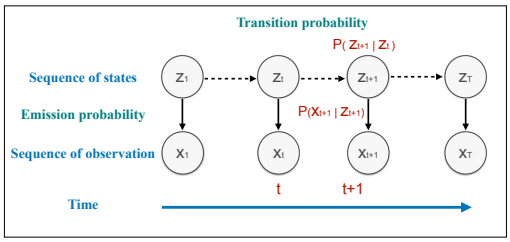


Figure 1: *The graphical model of HMM.*

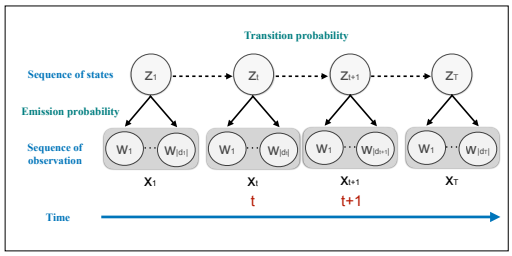


Figure 2: *The graphical model of HMMULM used to model the content of MOOCs.*

$$\gamma_t(i) = P(z_t = s_i|X, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{M}\alpha_t(j)\beta_t(j)} \quad (6)$$

$$\xi_t(i,j) = P(z_t = s_i, z_{t+1} = s_j|X, \lambda) = \frac{\alpha_t(i)A_{ij}\beta_{t+1}(j)\prod_{w\in V}B_j(w)^{c(w,x_{t+1})}}{\sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_t(i)A_{ij}\beta_{t+1}(j)\prod_{w\in V}B_j(w)^{c(w,x_{t+1})}} \quad (7)$$

In the M-step, the following equations are used to choose the parameters that maximize the likelihood of the observed sequence of lectures:

$$\pi(i) = \gamma_1(i) \quad (8)$$

$$A_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \quad (9)$$

$$B_i(w) = \frac{\sum_{t=1}^{T}\gamma_t(i)c(w,x_t)}{\sum_{v=1}^{V}\sum_{t=1}^{T}\gamma_t(i)c(v,x_t)} \quad (10)$$

For more information about Baum-Welch algorithm, please see [20]. It is clear that the E-step and M-step equations are very similar to the standard HMM except that instead of emitting one symbol, HMMULM emits one lecture represented as a bag-of-words.

5.4 Structural Topic Model

The Structural Topic Model (StrTM) [25] is another probabilistic graphical model that functions very similar to HMMULM. It has been used to model the latent topical structures inside documents. Like HMMULM, it models topics and their transitions as hidden states that emit lectures as bags-of-words. Unlike HMMULM, strTM assumes each lecture as a mixture of content topics and functional topic. Functional topic, denoted by z_B, is used to filter out document-independent words that models the corpus background (or general terms) [31]. Each word in the lecture is either generated by one of the content topics or the functional topic:

$$w \sim \theta P(w|\beta, z_i) + (1-\theta)P(w|\beta, z_B) \quad (11)$$

where θ is the controlling parameter. According to strTM, the probability of lecture x_j being generated by some topic z_i is:

$$P(x_j|z_i) = \prod_{w\in V}[\theta P(w|\beta, z_i) + (1-\theta)P(w|\beta, z_B)]^{c(w,x_j)} \quad (12)$$

Another difference between strTM and HMMULM, is that strTM assumes the transition probabilities $\mathbf{A}$ and the emission probability $\mathbf{B}$ are drawn from Multinomial distributions and use the conjugate Dirichlet distribution to impose a prior on the Multinomial distributions:

$$\alpha_z \sim Dir(\eta) \quad (13)$$

$$\beta_z \sim Dir(\gamma) \quad (14)$$

Where η and γ are the concentration hyper parameters that control sparsity of α_z and β_z respectively.

To estimate the parameters of strTM, we use the expectation-maximization algorithm as described by [25]. For more information about strTM, please refer to [25].

Table 1: *The dataset utilized in the experiment.*

Domain	# of Courses	# of Lectures	Avg # of Lectures
Python	21	460	22
SQL	15	247	16
ML	10	99	10

6. EVALUATION

In this section, we first demonstrate our dataset and the parameters settings. Second, we compare different models by studying the impact of topic transitions learned from various models on three lecture sequencing tasks. Finally, we qualitatively evaluate the topics and their transitions.

6.1 Dataset and Parameters Settings

We collected our dataset from real online courses using various MOOC platforms and in three different domains: Python, Structural Query Language (SQL), and Machine Learning Clustering algorithms (ML). Table 1 presents the statistic of the dataset. We use 75% of the data as a training set and 25% as a test set. To choose the number of topics in each domain, we manually inspected the dataset to choose the number of topics. The number of topics for Python, SQL, and ML were set to 13, 10, and 9 respectively.

Each course in the dataset is represented as a sequence of lecture video transcripts. We preprocess lecture transcripts by eliminating stop words and some rare terms. After cleaning the data, we constructed the bag-of-word vector representations of all lectures. We only use lecture transcripts to represent lectures; therefore, we only need to set two thresholds (K_1 and K_2) of the PCK-Means method in order to select the list of Must-link and Cannot-link constraints. Since we do not have labeled data we chose the thresholds that maximize

Table 2: *The performance of **Task 1**: Finding the correct sequence using the permutation method. It is clear that PCK-Means achieves the highest performance.*

Dataset	Measures	Methods				
		Cosine	PCK-Means	MULM	HMMULM	strTM
Python	kendall's $\tau(\sigma)$	0.60	**0.73**	0.49	0.66	0.54
	Dir-P	0.37	**0.50**	0.43	0.28	0.31
	Undir-P	0.52	**0.56**	0.50	0.44	0.35
	Lev-Sim	0.60	**0.63**	0.45	0.49	0.50
SQL	kendall's $\tau(\sigma)$	0.58	**0.59**	0.58	0.55	0.41
	Dir-P	**0.46**	0.43	0.40	0.36	0.41
	Undir-P	**0.67**	0.58	0.49	0.44	0.41
	Lev-Sim	0.53	**0.57**	0.54	0.49	0.37
ML	kendall's $\tau(\sigma)$	0.68	**0.75**	0.63	0.58	0.64
	Dir-P	0.34	0.34	**0.44**	0.34	0.43
	Undir-P	0.52	0.52	**0.57**	0.41	0.53
	Lev-Sim	0.61	**0.65**	0.55	0.43	0.53

the *Silhouette Coefficient* clustering measure using the training data. We set $K_1 = 0.55$ and $K_2 = 0.004$ for Python, $K_1 = 0.8$ and $K_2 = 0.01$ for SQL, and $K_1 = 0.55$ and $K_2 = 0.01$ for ML. To set the hyper parameters of strTM method, we used a grid search and chose the values that maximize the likelihood of the training data. We set $\theta = 0.2$, $\gamma = 0.3$, and $\eta = 0.6$ for Python, $\theta = 0.1$, $\gamma = 0.3$, and $\eta = 0.1$ for SQL, and $\theta = 0.1$, $\gamma = 0.1$, and $\eta = 0.6$ for ML.

6.2 Sequencing Tasks

In this experiment, our goal is to compare the topic transitions modeled by different methods in three tasks: 1) Finding the correct sequence of lectures, 2) Predicting the next lecture given a sequence of lectures, and 3) Predicting the sequence of a list of lectures where the first lecture in the sequence is given. An example of real application for task 1 and task 3 is designing a new course plan by sequencing lectures before delivering them to students. However, task 1 and task 3 exploit two different techniques to find the sequence. In contrast, task 2 can be applied to recommend the next lecture to learners to customize their learning based on the history of lectures they already watched. In the evaluation, the purpose of each task is to compare different methods and evaluate the ability of the parameters (A, B, and π) of each model to find the correct sequence in the three different tasks.

6.2.1 Evaluation Measures

To compare different models, we use the sequences of lectures from courses in the test set as the ground truth sequences and exploit different measures to do the evaluation. First, we follow Wang et al. [25] and use kendall's $\tau(\sigma)$. Kendall's $\tau(\sigma)$ is an information retrieval measure that captures the correlation between two ranked list. It indicates how the predicted order differs from the ground truth where 1 means perfect match, -1 means total mismatch, and 0 indicates that the two orders are independent. Second, we use Levenshtein normalized similarity which is the opposite of Levenshtein normalized distance that measures the minimum number of edits (insertions, deletions or substitutions) required to transform the predicted sequence to the ground truth sequence. The goal is to find the sequence that has the Levenshtein normalized similarity close to 1 which indicates that the number of edits required is minimal. Third, we utilize the directed bigram precision (see equation 15) that captures the correctness of the order between adjacent lectures. The intuition behind using this measure is to evaluate whether the transition maps learned by different models have the ability to capture the correct direction order between topics and adjacent lectures. Finally, we use the undirected bigram precision shown in equation 16 to measure whether the transition map of each model can recognize adjacent lectures but incorrectly captured the direction between topics.

$$P_{Dir-bigram} = \frac{\#\ of\ correct(a \rightarrow b)\ in\ estimated\ sequence}{\#\ of\ correct(a \rightarrow b)\ in\ ground\ truth} \tag{15}$$

$$P_{Undir-bigram} = \frac{\#\ of\ correct\{a,b\}\ in\ estimated\ sequence}{\#\ of\ correct\{a,b\}\ in\ ground\ truth} \tag{16}$$

6.2.2 Task 1: Finding The Correct Sequence

To find the correct sequence of lectures, we follow the permutation method utilized by [25]. With courses that have large number of lectures, it is infeasible to find all the orderings of lectures. Therefore, when the number of the permutations exceeds 500, we randomly permutated 500 possible orderings of lectures as candidates. We ran the experiment 20 times for each method and recorded the average results.

In order to select the optimal sequence from the list of permutations in strTM and HMMULM, we follow Wang et al. [25] and choose the sequence that has the highest generation probability calculated as:

$$\bar{\sigma}(m) = \arg\max_{\sigma(m)} \sum_{Z} P(x_{\sigma[0]}, x_{\sigma[1]}, \ldots, x_{\sigma[m]}, Z|\lambda) \tag{17}$$

To choose the best sequence for MULM, we first find the best topic **c** that generates each lectures in the test set according to equation 4. After that, we select the sequence that has the highest likelihood

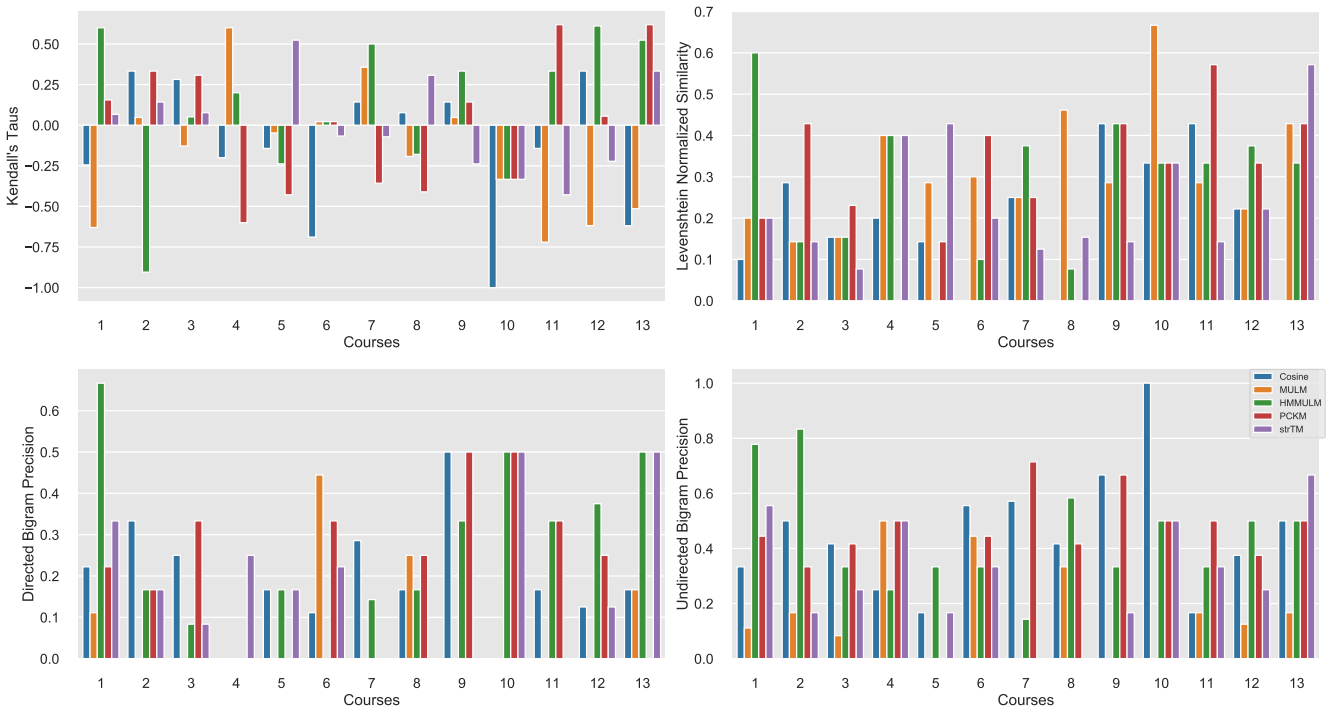


Figure 3: *The performance of different methods in **Task 3**: Predicting the whole sequence. All methods have comparable performance.*

based on the equation:

$$\begin{aligned}\bar{\sigma}(m) &= \arg\max_{\sigma(m)} P(C)P(X|C) \\ &= P(c_1)P(x_1|c_1)\sum_{i=2}^{|\sigma(m)|} P(c_i|c_{i-1})P(x_i|c_i)\end{aligned} \tag{18}$$

Since PCK-Means is a clustering method that minimizes the distance between lectures and clusters' centroids, we assign lectures x of the test set to the closest clusters z_i using Euclidean distances as shown in equation 19. Then, we select the sequence that maximizes the topic transitions between lectures in the sequence as well as minimizes the distance between adjacent lectures (see equation 20). The intuition behind that is to ensure the topic coherence between adjacent lectures and also reduces the gaps by minimizing the distance between them.

$$c(x) = \arg\min_{z_i} \|x - \mu_{z_i}\|^2 \tag{19}$$

$$\bar{\sigma}(m) = \arg\max_{\sigma(m)} \pi(c(x_1)) \sum_{i=2}^{|\sigma(m)|} A(x_{i-1}, x_i) - \|x_i - x_{i-1}\|^2 \tag{20}$$

As a baseline we accumulate the cosine similarity between adjacent lectures in the sequence and select the sequence in the permutations that has the highest similarity score to be the optimal sequence.

Table 2 summarizes the results of Task 1 for each method. We can notice that PCK-Means has the highest score in kendall's $\tau(\sigma)$ and Levenshtein normalized similarity in all datasets which indicates that PCK-Means has chosen the sequences that are very correlated to the ground truth sequences and need the minimal edits to be transformed to the ground sequences. However, PCK-Means only outperforms other models in the directed and undirected bigram precision in the Python dataset, indicating that it sometimes not able to capture the sequence between adjacent lectures.

Table 3: *The performance of **Task 2**: Predicting the next lecture. It is clear that HMMULM achienes the highest performance.*

	Accuracy		
Method	Python	SQL	ML
Cosine-Similarity	0.46	**0.56**	0.42
PCK-Means	0.45	0.49	0.47
MULM	0.41	0.34	0.37
HMMULM(Viterbi)	**0.52**	**0.56**	**0.60**
StrTM(Viterbi)	0.39	0.27	0.43

In general, it is clear that PCK-Means achieves the highest performance in most measures and almost in all the datasets. We think that combining the topic transitions with the Euclidean distance helps PCK-Means in finding the best sequence from the list of possible sequences.

6.2.3 *Task 2: Predicting The Next Lecture*

In task 2, each model predicts the next lecture given a sequence of lectures. We varied the length of the given sequence starting from one. As strTM and HMMULM are based on HMM, we utilized the Viterbi algorithm [20] to find the most probable sequence of hidden states or topics that generated the lectures in the given sequence. Then we greedily choose the next probable lecture in the sequence

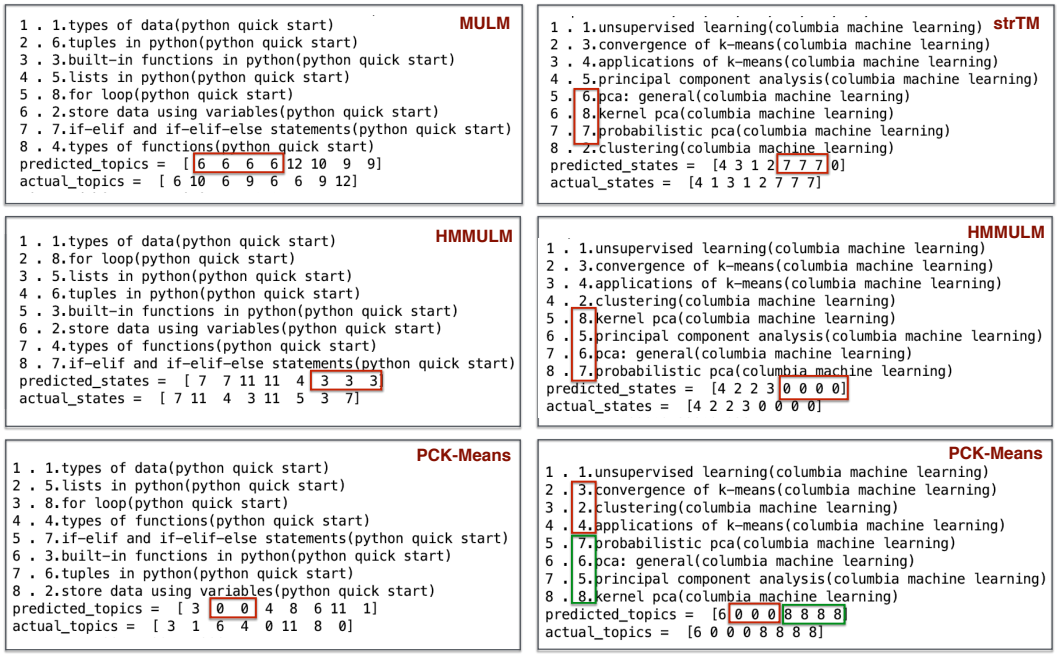


Figure 4: *Qualitative Analysis of Sequencing Task 3. (a) Examples of preferring self transition behaviour when selecting next lecture in the sequence, (b) Examples of the problem of sequencing adjacent lectures that cover the same topics.*

according to the equation:

$$\bar{x} = \arg\max_{x} P(z_i|z_{i-1})P(x|z_i) \tag{21}$$

Similar to task 1, for MULM and PCK-Means models we assign lectures to the best clusters using the equations 4 and 19 respectively. After that MULM greedily chooses the next lecture that maximizes the equation 21. On the other hand, PCK-Means model selects the next lecture that maximizes the topic transition and minimizes the distance with the last lecture in the given sequence. For the baseline, we use the cosine similarity where we choose the next lecture that has the highest similarity score with the last lecture in the given sequence.

Table 3 summarizes the results of Task 2 for each method. We can notice that HMMULM achieves the highest accuracy in all datasets. Using the Viterbi algorithm along with the learned topic transitions helps in capturing the most probable hidden states or topics that generate the given sequence of lectures. In addition, the topic transitions learned by HMMULM help in greedily pick the next lecture in the sequence. While StrTM also uses Viterbi algorithm similar to HMMULM, its accuracy scores were far less than HMMULM. We think the main reason for that due to the performance of the learned topic transitions as we explain in section 6.3.

6.2.4 Task 3: Predicting The Sequence

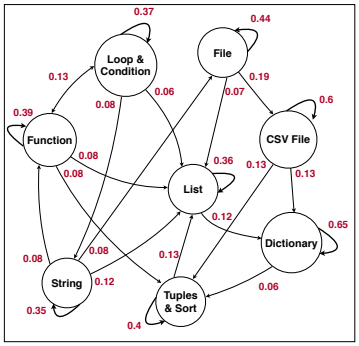
Task 3 is very similar to task 2 except that each method needs to find the whole sequence of given lectures where the first lecture in the sequence is given. Figure 3 depicts the results of Task 3.

As this task is considered the most challenging task, it is clear that there is no wining method. However, from the upper left graph that captures the Kendall's taus in Figure 3, we can notice that HMMULM has achieved a taus score ≥ 0.50 in four courses, PCK-Means has achieved the same score in only two courses, MULM and strTM in only one course, and Cosine method in non courses. For the Levenshtein normalized similarity, it is clear that all methods have comparable results. For the directed and undirected bigram precision, all methods have also comparable results except MULM. The reason is that MULM sometimes cannot complete the whole sequence because it only uses the greedy method which cannot complete the sequence in the case of the absence of the topic transitions required to sequence courses in the test set. In the case of other methods, they always find the whole sequence either because of the Viterbi algorithm used by HMMULM and strTM or due to the similarity or distance measures utilized by PCK-Means and Cosine methods.

In addition to quantitatively comparing the methods, we try to qualitatively evaluate the results by examining the generated sequences of each methods. In general, we found two common behaviour shared by all methods.

First, in most cases almost all the methods prefer self transition when they pick the next lecture in the sequence. For example, as shown in Figure 4 (a) , MULM, HMMULM, and PCK-Means select the next lecture that has the same topic as the current lecture.

Second, all methods cannot sequence lectures that belong to the same topic. In MOOCs, due to the short length of lectures, instructors sometimes explain the same topic using multiple lectures. As a result, it is hard to find the correct sequence of lectures that cover the same topic. For example, as shown in Figure 4 (b), the last four lectures of the course explain the "Principal Component Analysis algorithm" and hence strTM, HMMULM, and PCK-Means cannot predict the correct sequence of these lectures. In this case, we need to use other



Topics	Terms
String	string, index, quotes, strings, characters, quote, double, substring, character, eric, method.
Function	function, parameters, functions, arguments, return, parameter, formal, global, errors, def, scope.
File	file, files, open, directory, py, install, text, read, folder, line, lines.
CSV File	csv, phones, file, row, column, table, columns, delete, format, tabular, menu.
Loop & Condition	loop, true, false, condition, block, while, else, equal, statement, greater, boolean, infinite, elif.
Dictionary	dictionary, key, keys, dictionaries, pairs, counts, keyvalue, word, values, mapping, dict.
List	list, range, item, element, lists, index, elements, loop, sequence, items, strings.
Tuples & Sort	tuple, sorted, random, sort, tuples, sorting, list, reverse, lambda, function, order.

Classes	class, attributes, object, methods, teaching, classes, method, language, programming, objects, instance.
Prog. Language	programming, language, concepts, machine, hours, learn, tasks, tool, videos, calculations, languages.
List	list, accumulator, state, num, item, append, accumulate, items, object, position, mutate.
Not Clear	expression, area, largest, guess, recursive, root, expressions, divided, floating, pi, cube.
Not Clear	split, api, preview, join, space, image, white, comma, format, file, words.

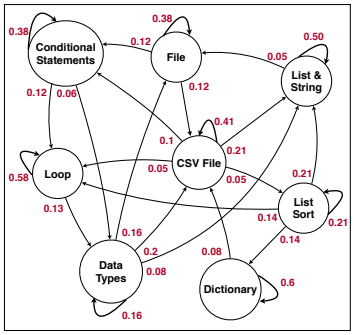
Figure 5: *Python topics and their transitions using **PCK-Means** method. The table represents the top terms in each topic.*

techniques to predict the sequence. One naive solution is to assume that all adjacent lectures the belong to the same topic as one atomic unit and we only need to sequence lectures that have different topics. Further investigation of solving the sequencing problem of lectures that belong to the same topic is left for future work.

6.3 Topic Transitions Examples

In this section, we present examples of topics and topic transitions learned by different methods. Due to space constraint, we present examples using Python dataset. We try to analyze the words with the highest probabilities in the word distributions of topics learned by each methods and manually mapped them to topic words or phrases. For instance, if the word distribution has the words: *list, range, items, index, and append*, then it is clear that this word distribution captures the topic "List". The word distributions with topic phrases of each topic learned by PCK-Means, MULM, HMMULM, and strTM methods in Python dataset are depicted in Figure 5, 6, 7, and 8 respectively. Since we have 13 topics in the Python dataset, we only visualize the topic transitions of a subset of these topics and depicted the transitions that have scores ≥ 0.05.

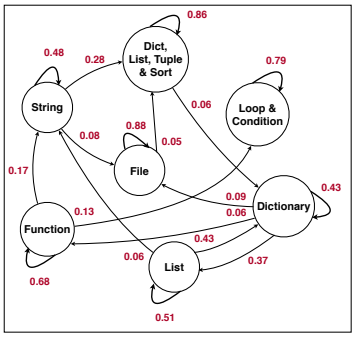
It is clear from the Figures that all models extract some useful topics where the top terms of each topic clearly explain the topic. However, PCK-Means has the best word distributions that clearly explain each topic followed by HMMULM and then MULM while strTM has the lowest performance. We also notice from the Figures that PCKMeans have extracted 11 useful topics with two topics that have unclear word distributions and cannot be mapped to any useful topics. In contrast, MULM has modeled 10 meaningful topics with three topics form noise and cannot be mapped to any topics. On the other hand, HMMULM and strTM capture 9 topics with four unclear topics that cannot be mapped to any phrase. In general, this finding indicates that PCK-Means has the best performance in modeling the topics of the courses in the Python dataset as it models more useful topics with clear word distributions. The results also indicate that strTM achieves the lowest performance because even though it captures the same number of meaningful topics as HMMULM, strTM has the lowest performance in the clarity of the word distributions.



Topics	Terms
CSV File	csv, quotes, file, string, format, row, quote, column, double, table, comma, columns.
File	file, function, open, lines, line, area, seconds, files, read, object, close.
Loop	loop, largest, while, infinite, loops, smallest, break, statement, blah, true, iteration.
Data Types	float, expression, integer, floating, type, int, types, floats, expressions, convert, decimal.
Conditional Statements	true, false, block, else, condition, equal, expression, greater, statement, username, elif, boolean.
Dictionary	dictionary, key, keys, dictionaries, pairs, counts, keyvalue, list, word, tuple, values.
List Sort	sorted, random, sorting, sort, list, programming, accumulation, language, loop, lost, lesson.
List & String	list, index, object, string, method, methods, class, item, position, strings, items.

Installing Python	file, install, directory, py, command, tool, windows, files, window, click, folder, ipython.
Prog. Language	programming, language, learn, concepts, tasks, scripts, practice, learning, videos, script, feel.
Not Clear	accumulator, list, guess, state, root, function, square, cube, dictionary, count, total.
Not Clear	turtle, floor, api, random, comments, calculations, module, program, errors, stack, machine.
Not Clear	list, tuple, function, element, range, loop, global, recursive, tuples, sequence, parameter.

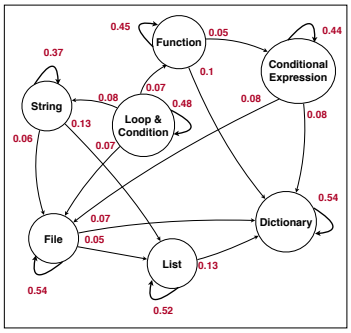
Figure 6: *Python topics and their transitions using **MULM** method. The table represents the top terms in each topic.*



Topics	Terms
String	string, index, character, strings, quotes, list, substring, characters, preview, location, slice.
Function	function, float, type, expression, return, parameter, parameters, integer, int, parentheses, string.
File	file, csv, files, phones, row, format, open, column, read, table, directory.
Loop & Condition	loop, true, false, while, condition, loops, equal, largest, block, infinite, else.
Dictionary	dictionary, list, accumulator, key, item, keys, count, counts, loop, state, num.
List	list, object, append, lists, items, position, lst, reference, strings, mutate, bound.
Dict, List, Tuple & Sort	dictionary, list, tuple, key, sorted, function, keys, sort, expression, item, sorting.

Classes	class, attributes, methods, instance, dictionary, method, turtle, classes, module, objects, concepts.
Prog. Language	language, programming, learn, script, data, learning, syntax, tasks, concepts, languages, scripts.
Not Clear	teaching, eric, code, cell, grade, install, button, messages, errors, error, month.
Not Clear	comments, calculations, floor, input, algorithms, computers, warm, web, city, hot, fast.
Not Clear	seconds, block, traceback, hours, blah, turtle, line, code, runs, times, execute.
Not Clear	kelvin, temperature, hours, fahrenheit, equal, program, step, error, loop, celsius, indent.

Figure 7: *Python topics and their transitions using **HM-MULM** method. The table represents the top terms in each topic.*



Topics	Terms
String	list, string, if, position, index, python, strings, quotes, character, number, single, type.
Function	function, return, code, parameter, expression, square, inside, functions, print, parameters, statement, arguments.
File	file, csv, files, open, line, read, list, row, lines, data, quotes, table.
Loop & Condition	if, loop, code, while, run, else, statement, block, equal, condition, true.
Dictionary	dictionary, key, list, function, keys, dictionaries, sorted, lists, sort, tuple, values.
List	list, lists, element, item, index, loop, function, if, count, strings, file.
Conditional Expression	true, if, false, else, equal, expression, code, boolean, block, greater, expressions, operators.

List & Dict	list, character, dictionary, item, loop, accumulator, variable, number, count, index, items.
Classes	list, class, tuple, function, method, set, data, tuples, index, methods, object.
Not Clear	loop, if, true, python, statement, largest, code, false, run, smallest, variable.
Not Clear	python, variable, string, type, if, integer, point, variables, code, floating, kind.
Not Clear	if, loop, guess, root, machine, start, simply, number, answer, times, cube.
Not Clear	data, dictionary, language, kelvin, keys, programming, table, row, column, dictionaries, columns.

Figure 8: *Python topics and their transitions using **strTM** method. The table represents the top terms in each topic.*

As shown in the Figures 5, 6, 7, and 8, the meaningful topics extracted by all methods are very similar with some variations. For example, while PCK-Means and MULM separate "*Files*" and "*CSV Files*", HMMULM and strTM combines them into one topic. In addition, while PCK-Means and HMMULM combines "*Loops* & *Condition*", MULM differentiates between them. StrTM, on the other hand, has both "*Loops* & *Condition*" and "*Conditional Statements*."

Regarding the topic transitions, it is clear that all models capture self transitions with the topic and itself. This indicates that, in MOOCs, instructors used multiple lectures to explain the same topic. However, HMMULM gave higher probability to self transitions compared to other methods. We can notice from the Figures that there are some consensus between all methods on some transitions between topics such as: "*List*"→ "*Dictionary*" and "*String*"→ "*File*." There are also some variations of topic transitions between different models. For instance, wile PCK-Means, HMMULM, and strTM have a transition between "*String*"→ "*List*", MULM combines these two topics into one topic or cluster. Another variation is that, PCK-Means, strTM and MULM have a transition "*Loop* & *Condition*"→ "*String*", whereas HMMULM misses this transition.

In general, all methods captures useful topics with clear word distributions. Regarding the topic transitions, all methods capture self transitions and also have some consensus on some transitions. There are also some variations between methods and these differences due to how each method identify topics of each lecture. Improving the modeling of topics and the mapping between lectures and topics clearly would improve the quality of the topic transition maps.

7. CONCLUSION

In this paper, we introduce the Topic Transition Map which is a general structure that models the content of MOOCs as topics, where each lecture is mapped to a topic, and captures the transition between topics. It models the various ways of how instructors organize topics in order to construct the study plan of their courses. We investigate four different methods to construct the Topic Transition Map: PCK-Means, MULM, HMMULM, and strTM. PCK-Means and MULM separately cluster lectures into topics and then learn the transitions between topics, by leverage the sequences of lectures in different courses. In contrast, HMMULM and strTM assume first order Markov property among latent topics and hence jointly learn topics and their transitions. While the three model, MULM, HMMULM, and strTM are probabilistic models, PCK-Means is distance-based clustering algorithm that incorporates some constraints to guide the clustering process.

We evaluated the generated topic transitions from various methods using three different tasks: 1) determining the correct sequence, 2) predicting the next lecture, and 3) predicting the sequence of lectures. Our evaluation revealed that PCK-Means achieves the highest performance in determining the correct sequence while HMMULM outperforms other methods in the task of predicting the next lecture. Since the task of predicting the whole sequence of lectures is considered the most challenging task, there was no winning method and all methods have comparable performance with MULM has the lowest performance as it sometimes fails to predict the whole sequence. We also visualize the the Topic Transition Maps generated by different methods to qualitatively evaluate the resulted maps. We found that PCK-Means has extracted more meaningful topics with the best word distributions that clearly explain each topic.

In the future, we plan to explore incorporating Topic Transition Map with concept dependency relations and examine if this can solve the problem of sequencing lectures that belong to the same topic. Further, we aim to combine different methods such as PCK-Means and HMMULM in order to improve the accuracy of the Topic Transition Map and hence improving the performance of the sequencing tasks. Finally, we plan to apply our work on other domains such as traditional University courses or educational books. To do that, we need to investigate how to divide long lectures or book sections into segments where each segment is mapped to one topic.

8. REFERENCES

[1] R. Agrawal, B. Golshan, and E. Papalexakis. Toward data-driven design of educational courses: A feasibility study. *JEDM-Journal of Educational Data Mining*, 8(1):1–21, 2016.

[2] F. ALSaad and A. Alawini. Unsupervised approach for modeling content structures of moocs. In *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, pages 18–28, 2020.

[3] F. ALSaad, A. Boughoula, C. Geigle, H. Sundaram, and C. Zhai. Mining mooc lecture transcripts to construct concept dependency graphs. In *Proceedings of the 11th International Conference on Educational Data Mining*, pages 467–473. EDM, 2018.

[4] S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 333–344. SIAM, 2004.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[6] D. Chaplot and K. R. Koedinger. Data-driven automated induction of prerequisite structure graphs. In *Proceedings of the 9th International Conference on Educational Data Mining*, pages 318–323. EDM, 2016.

[7] W. Chen, A. S. Lan, D. Cao, C. Brinton, and M. Chiang. Behavioral analysis at scale: Learning course prerequisite structures from learner clickstreams. In *Proceedings of the 11th International Conference on Educational Data Mining*, pages 66–75. EDM, 2018.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39 (1):1–22, 1977.

[9] L. Du, J. K. Pate, and M. Johnson. Topic models with topic ordering regularities for topic segmentation. In *2014 IEEE International Conference on Data Mining*, pages 803–808. IEEE, 2014.

[10] L. D. Fink. *Creating significant learning experiences: An integrated approach to designing college courses*. John Wiley & Sons, 2013.

[11] R. M. Gagne and L. J. Briggs. *Principles of instructional design.* Holt, Rinehart & Winston, 1974.

[12] A. Gruber, Y. Weiss, and M. Rosen-Zvi. Hidden topic markov models. In *Artificial intelligence and statistics*, pages 163–170. PMLR, 2007.

[13] T. Hofmann. Probabilistic latent semantic analysis. *arXiv preprint arXiv:1301.6705*, 2013.

[14] C. Liang, J. Ye, Z. Wu, B. Pursel, and C. L. Giles. Recovering concept prerequisite relations from university course dependencies. 2017.

[15] H. Liu, W. Ma, Y. Yang, and J. Carbonell. Learning concept

graphs from online educational data. *Journal of Artificial Intelligence Research*, 55:1059–1090, 2016.

[16] R. Manrique, J. Sosa, O. Marino, B. P. Nunes, and N. Cardozo. Investigating learning resources precedence relations via concept prerequisite learning. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 198–205. IEEE, 2018.

[17] L. Pan, C. Li, J. Li, and J. Tang. Prerequisite relation learning for concepts in moocs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1447–1456, 2017.

[18] A. Parameswaran, H. Garcia-Molina, and A. Rajaraman. Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment*, 3(1-2):566–577, 2010.

[19] G. Pfundstein. *Hidden markov models with generalised emission distribution for the analysis of high-dimensional, non-euclidean data*. PhD thesis, Institut für Statistik, 2011.

[20] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[21] R. Scheines, E. Silver, and I. M. Goldin. Discovering prerequisite relationships among knowledge components. In *EDM*, pages 355–356, 2014.

[22] S.-s. Shen, H.-y. Lee, S.-w. Li, V. Zue, and L.-s. Lee. Structuring lectures in massive open online courses (moocs) for efficient learning by linking similar sections and predicting prerequisites. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[23] A. Vuong, T. Nixon, and B. Towle. A method for finding prerequisites within a curriculum. In *EDM*, pages 211–216, 2011.

[24] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001.

[25] H. Wang, D. Zhang, and C. Zhai. Structural topic model for latent topical structure analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1526–1535, 2011.

[26] S. Wang, A. Ororbia, Z. Wu, K. Williams, C. Liang, B. Pursel, and C. L. Giles. Using prerequisites to extract concept maps fromtextbooks. In *Proceedings of the 25th acm international on conference on information and knowledge management*, pages 317–326, 2016.

[27] K. WAUTERS, P. DESMET, and W. VAN DEN NOORTGATE. Acquiring item difficulty estimates: a collaborative effort of data and judgment. In *EDM 2011 4 th International Conference on Educational Data Mining*, page 121.

[28] J. Xu, J. Liu, and K. Araki. A hybrid topic model for multi-document summarization. *IEICE TRANSACTIONS on Information and Systems*, 98(5):1089–1094, 2015.

[29] Y. Yang, H. Liu, J. Carbonell, and W. Ma. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 159–168. ACM, 2015.

[30] B.-J. Yoon. Hidden markov models and their applications in biological sequence analysis. *Current genomics*, 10(6): 402–415, 2009.

[31] C. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 743–748, 2004.

[32] S. Zheng, M. B. Rosson, P. C. Shih, and J. M. Carroll. Understanding student motivation, behaviors and perceptions in moocs. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*, pages 1882–1895. ACM, 2015.