Enhancing Personalized Learning of students through Study Material Recommendation in an Adaptive Learning Environment.

Madushan Pathirana – 2020BA024

Supervisor – Dr. L.N.C De Silva

Table of Content

1 Data

This research uses a real-world data set from an International E-learning (courseware) platform that uses state of the art adaptive learning technology. This platform provides educational content targeting schools for Mathematics, Economy, Chemistry, Biology, Physics and Psychology. Based on the research question, identified data was already collected with the organization's approval.

Subjected Adaptive Learning Platform (ADP) measures the learners' progress level ranging from 0 to 100. Teachers can assign assignments to the student related to a specific Learning Objective(LO). A student has to reach 100 progress to complete the assignment, then the student has achieved the 'Mastery' to that LO. Each LO has minimum 4 question, progress of a student for a given LO is

Progress = proficiency score x fraction of the minimum questions learner have tried

If student fail master a LO, student get to do more practice questions. If the student need further support, he or she get more instructions and direct back to the prerequisite LOs.

All the learning objectives, concepts, questions, and course materials are associated to knowledge graphs. These knowledge graphs and progress levels drive the students journey to master a given learning objective. But other characteristics of the student joinery are not considered. Such as time spent on a question, time spent on instructions, quality of the instruction materials, etc.

Data	Number of data points	Attributes
Student coursework performance	3.3 million	• Learning objectives • coursework id • user id • progress • question id • correctness of the answer • time spent to answer

		• time spent for the question instruction • study material id referred
Student assignment	140,000	• Learning objectives • test id • user id • question id • correctness of the answer
Learning objective map (knowledge graph)	1145	• Source LO Id (prerequisite LO ID) • Destination LO Id • Source LO Title (prerequisite LO Name) • Destination LO Title

2 Solution design

2.1 Selection of solution architecture

According to the literature authors have used different methods to solve knowledge tracing. There mainly two methods. First method is Traditional knowledge tracing which has two branches. They are;

1) Bayesian knowledge tracing
2) Factor analysis models

Second method is Deep knowledge tracing. This is the latest knowledge tracing methodology, and it has outperformed Traditional knowledge tracing methods. Our dataset has already tested with modified item response theory which is one of the models under Factor analysis models. Hence Traditional knowledge tracing methods will not be used for this research. Instead, Deep knowledge tracing methods will be employed expecting better performance.

Under Deep knowledge tracing there are multiple models. All these models use Deep Neural Networks with different input types and different neural network architecture. Subjected data set has heterogenous data types and relationship between these data better explained by Graphs/Networks. Hence this study will use Graph based knowledge tracing methodology to predict students' knowledge level. There are multiple graph based knowledge tracing methods in literature and this study will compare and contrast different model when building the model.

2.2 Graph Neural Network (GNN)

The evolution of Graph Neural Networks (GNNs) has given rise to numerous applications across diverse domains, including but not limited to Natural Language Processing (NLP) Computer Vision (CV), and Recommendation Systems . GNNs, with their capacity to capture high-order information, have paved the way for substantial advancements in these fields. In our research, we harness the power of Graph Convolutional Neural (GCN) within our Graph-based Interaction Knowledge Tracing (GIKT) model. By employing GCN, we aim to extract meaningful relations between skills and questions, effectively translating them into rich and informative representations. As far as our knowledge extends.

2.3 How GNN works

Graph Neural Networks (GNNs) are a type of deep learning model designed to work with graph-structured data, where data is organized as nodes connected by edges (like a social network, a road map, or a recommendation system). GNNs aim to understand and process this data effectively.

Each node in the graph starts with an initial representation, typically as a vector of numbers. These initial representations capture the characteristics of each node.

GNNs operate through a process of message passing. At each step, each node sends messages to its neighboring nodes. These messages typically contain information about the node itself and its immediate neighbors. The idea is that nodes can exchange information and learn from each other.

After receiving messages from their neighbors, nodes aggregate this information to update their own representation. This aggregation process combines the information from the node itself with

that of its neighbors. This is done by Neural Networks. In the below Figure 2.1 gray boxes show the neural networks.

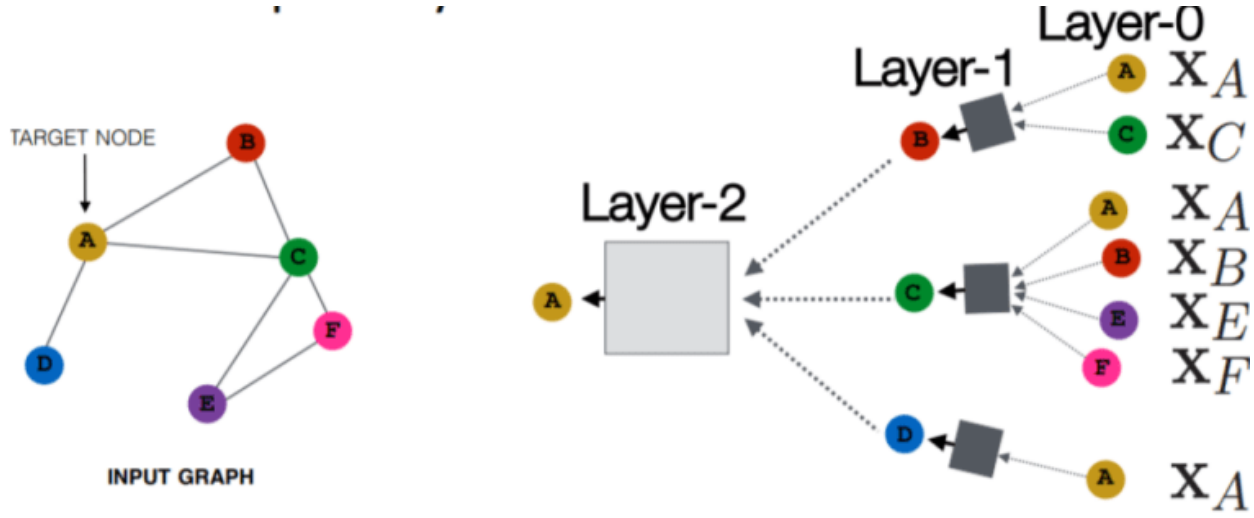

Figure 2.1 - Graph Neural Network (source - Stanford Graph based Machine Learning lecture slides)

For example, (X_A) is a feature vector of node A. The inputs are those feature vectors, and the box will take the two feature vectors (X_A and X_c), aggregate them, and then pass on to the next layer.

$$h_v^0 = X_v \ (feature\ vector)$$

Equation 2.1 - feature vector

Notice that, for example, the input at node C are the features of node C, but the representation of node C in layer 1 will be a hidden, latent representation of the node, and in layer 2 it'll be another latent representation. At each k^{th} layer, h_v^k feature vector produced by the Equation 2.2. It average the previous layer by the number of nodes in the current layer and add bias to previous layer, then perform a nonlinear activation denoted by σ. W_k (weight matrix) and B_k (bias matrix) are trainable parameters.

$$h_v^k = \sigma(W_k \sum \frac{h_u^{k-1}}{\mid N(v)} + B_k h_v^{k-1}) \text{ where } k = 1, ..., k-1$$

Equation 2.2 - Neighborhood aggregation

2.4 Risk

Currently notified risk is not having enough hardware capacity to perform computations. This risk could be overcome by batch wise processing.