A Graph Neural Network Model for Concept Prerequisite Relation Extraction

Debjani Mazumder
debjani22.mazumder@iitkgp.ac.in
IIT, Kharagpur
India

Jiaul H. Paik
jiaul@cet.iitkgp.ac.in
IIT, Kharagpur
India

Anupam Basu
anupambas@gmail.com
Sister Nivedita University
Kolkata, India

ABSTRACT

In recent years, with the emergence of online learning platforms and e-learning resources, many documents are available for a particular topic. For a better learning experience, the learner often needs to know and learn first the prerequisite concepts for a given concept. Traditionally, the identification of such prerequisite concepts is done manually by subject experts, which in turn, often limits self-paced learning. Recently, machine learning models have found encouraging success for the task, obviating manual effort. In this paper, we propose a graph neural network based approach that leverages node attention over a heterogeneous graph to extract the prerequisite concepts for a given concept. Experiments on a set of benchmark data show that the proposed model outperforms the existing models by large margins almost always, making the model a new state-of-the-art for the task.

CCS CONCEPTS

• **Information systems** → **Information extraction**; • **Computing methodologies** → Neural networks.

KEYWORDS

Heterogeneous Graph, Graph Neural Network, Graph Attention Network, Pedagogical Concepts, Relation Extraction

ACM Reference Format:
Debjani Mazumder, Jiaul H. Paik, and Anupam Basu. 2023. A Graph Neural Network Model for Concept Prerequisite Relation Extraction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614761

1 INTRODUCTION

The curriculum is an important component in every pedagogical system, from a single day instructional session to a comprehensive four year degree program. For a specific class, multiple topics come together to create a balanced curriculum. In recent years, with the emergence of online learning platforms and e-learning resources, many documents are available for a particular topic. The learners often face difficulties in choosing the right learning resources in the right order that helps towards a good learning experience. Typically, a learning resource explains one or more *concepts* (the central concepts) and at the same time contains other related concepts to the central concepts. In this article, we refer to concepts as the salient words in a textual learning resource. Thus, our focus is on textual content rather than other types of learning resources.

For ease of learning and better outcomes, concepts need to be understood in the right order. Usually, the order of learning is determined by the relationship between the concepts. One such important relationship between concepts is the prerequisite relationship. A concept C_1 is considered a prerequisite concept for C_2 if the understanding of C_2 requires the knowledge of C_1. Thus, in our work, given a concept pair (C_i, C_j), we categorize them into one of the three types: i) *prerequisite concept*, the concept that needs to be learned before learning another concept, ii) the *outcome concept*, which can be learned after learning the prerequisite concept and iii) no such relationship exists. For example, students need to learn *stack* before learning *depth-first search*. Thus, *stack* is a prerequisite concept for *depth-first search*. On the other hand, the prerequisite-outcome relationship does not exist between the concept pair (*stack* and *vector*). Prerequisite concept identification has several downstream applications. Some well known areas where this is used are intelligent tutoring [8, 44], curriculum planning [1, 29], and the automatic sequencing of learning materials [6].

Identification of prerequisite concepts for a given concept has been attempted using various types of methods, such as a simple link based approach, classical ML models with hand crafted features, and, more recently, deep learning models. Since the underlying problem can be naturally modeled as a graph, graph neural networks have been widely used in the recent past. These models indeed improve the performance on this task considerably compared to the traditional models. Relational graph convolutional network, graph autoencoder, and multi-headed attention on the graph are some of the most recent approaches that have been found to be very effective. The link based approaches are simple and efficient but do not provide competitive performance to the supervised machine learning models. The deep learning models improve state-of-the-art further by finding automatically customized concept features for prerequisite relation prediction.

In this paper, we propose a deep learning model based on a graph attention network over a heterogeneous document-concept property graph. In our graph, we add documents as well as the concepts in the documents as graph nodes. We add three types of edges to encode different relationships between nodes in the graph. The graph also has edge weights, and it depends on the relationship between the two incident nodes on the edge. We incorporate node features that consist of dense vector representations representing either the concept or the document associated with the node. We then

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3614761

use a graph attention network over the heterogeneous property graph. The primary objective is to generate focused node embedding that will aid in identifying the desired relationship between two concepts. The attentional node embeddings are then used in another simple network (called a prediction network) to produce the final class label for the edge. The graph attention network and the prediction network are trained jointly to allow the network to update the model parameters appropriately and collaboratively. We conduct a set of experiments on four well-known benchmark datasets and compare the performance of the proposed method against a set of 12 existing baselines. Our model consistently and significantly outperforms the existing models in terms of Precision (P), Recall (R), and F_1 score.

2 RELATED WORK

Inferring prerequisite relations between pedagogical concepts has attracted considerable attention in the recent past. We provide an overview of different connections between concepts, followed by a discussion of existing approaches for learning concept prerequisite relations. Grefenstette [14] applied the Jaccard Similarity measure to relate concepts with each other. White et al. [46] suggests a correlation based measure for topic similarity where they have studied 10 similarity metrics on a small sample of 10 pairs of topics. Ho et al. [16] proposed a hierarchical learning method that utilizes ACL anthology and Wikipedia to learn the topic taxonomy from them. For a topic modeling approach performed over document citation links rather than over words or n-grams, Wang et al. [45] proposed a method for topic A's dependence on another topic B as the probability of a document in A citing a document in B.

Sayyadiharikandeh et al. [38] clickstream on Wikipedia for feature definition, followed by a binary classifier to infer concept prerequisite relations. Reference Distance (RefD) [24] uses link based metrics and a concept graph to model technical concepts and relations, utilizing Cross-Entropy [13] and the Information-Flow method to reveal prerequisites. CPR-Recover [26] is unsupervised, recovering relations from course dependencies and concept graph sparsity. Talukdar et al. [42] use Wiki features and a maximum entropy classifier to extract prerequisites from text and crowd-generated data. Pan et al. [35] propose a learning-based approach with contextual, structural, and semantic features for relation classification. Roy et al. [37] introduce PREREQ, employing pairwise-link LDA [33] for concept representation and training on a Siamese network for prerequisite determination.

A Variational Graph Autoencoder (VGAE) is a model that uses graph networks to predict concept prerequisite relations based on a concept graph [23]. Nevertheless, this approach overlooks intricate relationships among concepts and educational documents and demands a substantial amount of training data. Active learning strategies were explored to enhance concept prerequisite learning, aiming to lower the burden of collecting extensive training data [25, 28]. Li et al. [22] introduced the relational variational graph autoencoder (R-VGAE) to predict relations among concept nodes within a graph that comprises both concept and resource nodes. In a similar vein, Liu et al. [29] proposed diverse approaches, including classification, learning to rank, and nearest neighbor search, to ascertain prerequisite relations at both course and concept levels through two level directed graphs. From a different perspective, Lu et al. [30] initially identified domain specific concepts using a graph based ranking method and subsequently introduced the iPRL method. This iterative framework integrates learning-based and recovery-based models for identifying prerequisite relations among concepts. CPRL [18] presents a concept prerequisite learning strategy applied to a heterogeneous concept learning-object graph. They harnessed an RGCN [40] model to portray concept features, succeeded by a Siamese network for predicting prerequisite relations. On a similar note, Zhang et al. [48] identified concept prerequisite relations within a resource-concept graph utilizing multi-head attention and a variational graph autoencoder (MHAVGAE).

However, each of these existing approaches comes with its own limitations. Some models treat all neighboring nodes as equally significant contributors to a specific node in the resource-concept graph. For instance, let us consider the nodes *linked-list*, *pointer*, and *stack* in a concept graph. For the node *linked-list*, *pointer* should hold greater significance compared to *stack*. Additionally, certain frameworks create resource and concept graphs separately, leading to increased memory usage and reduced cost efficiency. Our proposed approach addresses these challenges associated with establishing prerequisite relations among educational concepts within a document-concept graph.

One of the main problems with traditional machine learning methods is that they require handcrafted features to process unstructured data like graphs. Sperduti et al. [41] first applied neural networks to directed acyclic graphs. A graph based hierarchical clustering approach to learn hierarchies from both structured and unstructured data was proposed by Joyner et al. [19]. The notion of graph neural networks was primarily outlined in Gori et al. [31] and further elaborated in Scarselli et al. [39], and Gallicchio et al. [11]. Recurrent graph neural networks (RecGNNs) fall into these early studies. Bronstein et al. mention bronstein2017geometric comprehensively studied several GCN methods and CNNs on manifolds through deep geometric learning. Battaglia et al. [4] utilized GNNs and GCNs for relational reasoning using a graph neural network.

Graph Convolutional Networks (GCN) [21] offer scalable semi-supervised learning for unstructured data like graphs, finding applications in recommendation systems [12, 49], traffic prediction [10], image recognition [9], and more. Various GCN variants tackle tasks such as classification, link prediction, entity relations, and influence maximization [15]. Kopf et al. [20] proposed a variational graph autoencoder for link prediction, while Schlichtkrull et al. [40] introduced relational graph convolutional network (R-GCN) for link prediction and entity classification in massive relational data.

Chen et al. [7] presented the multi-level Graph Convolutional Networks, which employ both local and hypergraph level graph convolutions to learn network embeddings for anchor link prediction. Dealing with the complexity and noise in real-world graph structures has prompted various solutions. For example, Wu et al. [47] enhance classification using multiple views of the same graph. Another emerging approach is incorporating *attention* into graph-structured data. Attention, introduced by Bahdanau et al. [3, 32], helps models focus on relevant data parts. The graph attention network, proposed by Velickovic et al. [43], is an attention based model that prioritizes significant neighbor nodes.

3 PROPOSED WORK

The main objective of this work is to uncover the prerequisite relations among the concepts that are present in learning or pedagogical resources. The pedagogical resources can be book chapters, university courses, or web documents and are in textual form. We assume that the concepts are already marked in the learning resources, and thus concept spotting from the documents is not part of our work. We only focus on prerequisite relation identification. Towards that goal, we propose a model based on a graph attention neural network that leverages a property graph over concepts and the documents containing these concepts. Thus, the proposed model has two main components: i) the graph construction and ii) the neural model on the graph. Furthermore, the second component has two sub-networks. The first sub-network uses a graph attention network on the property graph for node representation, while the second sub-network uses the learned representations for predicting the prerequisite relation label. The two sub-networks are trained jointly. In other words, the parameters of both networks are updated simultaneously based on the prediction error.

3.1 Problem Definition

A corpus is a collection of educational documents and their related concepts. In the same subject area, n number of documents is denoted as $\mathcal{D} = \{d_1, ..., d_i, ...d_n\}$, where d_i is one document. Every document contains one or more concepts. Concepts play an important role in learning a specific document. For a corpus of documents $\mathcal{D}$, a set of m concepts in $\mathcal{D}$ is expressed as $C = \{c_1, c_2..., c_m\}$.

Formally, the precise goal of the proposed method is as follows. Given an academic resource $\mathcal{D}$ and its corresponding set of concepts C, the objective is to learn a function $\mathcal{F}_\theta : C \times C \rightarrow \{0, 1\}$, which can determine if c_i is a prerequisite concept of c_j. We cast this as a binary classification task. Specifically, if $\mathcal{F}_\theta(c_i, c_j) = 1$, then c_i is the prerequisite concept of c_j, otherwise c_i is not a prerequisite concept of c_j. It is important to note that the $\mathcal{F}_\theta(.,.)$ is not symmetric. That means, it may happen that $\mathcal{F}_\theta(c_i, c_j) = 0$, but $\mathcal{F}_\theta(c_j, c_i) = 1$. In that case, c_j is the prerequisite for c_i, but c_i is not a prerequisite concept of c_j.

3.2 Graph Construction

A set of educational documents and its corresponding concepts is used to construct a heterogeneous graph $\mathbf{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ consists of two types of nodes: the concept nodes and the document nodes.

Our heterogeneous graph $\mathbf{G}$ comprises of the following three types of edges:

(1) **Document-to-concept:** The edge between the document vertex and the concept vertex reflects the importance of a concept in a document. This edge is denoted by e_{dc}.
(2) **Concept-to-concept:** This edge represents the relatedness of two concepts. Specifically, this is measured by how often two concepts co-occur in a fixed-sized window. The edge between two concept vertices is expressed by e_{cc}.
(3) **Document-to-document:** This edge connects two documents based on their similarity of content. The primary role of this edge is to improve the representation of documents, which can potentially improve the concept representation. We denote the edge between two document vertices by e_{dd}.

In the heterogeneous graph $\mathbf{G}$, $\mathcal{V}_d \in \mathbb{R}^{\mathcal{N}_d \times \mathcal{F}}$ and $\mathcal{V}_c \in \mathbb{R}^{\mathcal{N}_c \times \mathcal{F}}$ represent the features of document and concept vertices, and $A_{dd} \in \mathbb{R}^{\mathcal{N}_d \times \mathcal{N}_d}$, $A_{dc} \in \mathbb{R}^{\mathcal{N}_d \times \mathcal{N}_c}$, $A_{cc} \in \mathbb{R}^{\mathcal{N}_c \times \mathcal{N}_c}$ are the values of the three types of edges between these vertices. $\mathcal{N}_d$ and $\mathcal{N}_c$ are the number of document vertices and concept vertices, respectively. $\mathcal{F}$ is the feature dimension of these vertices. We introduce the details of these feature extraction in Section 4.2.

Given a set of educational documents, concept pairs (c_i, c_j), vertex features $\mathcal{V}_d$ and $\mathcal{V}_c$ along with all the edges e_{cc}, e_{dc} and e_{dd} in the document-concept graph, we want to find an objective function $\mathcal{F}_\theta$ that will determine the following:

$$\mathcal{Y}(c_i, c_j) = \begin{cases} 1, & if\ c_i\ is\ a\ prerequisite\ of\ c_j \\ 0, & otherwise. \end{cases} \tag{1}$$

Graph Properties

We reiterate that the graph we are dealing with has two types of properties: i) the node properties and ii) the edge properties. We next detail the approach for generating the properties.

Vertex properties

The concept-document property graph contains two types of nodes: i) the concept nodes and ii) the document nodes. In order to create the properties for the concept nodes, we utilize the co-occurrence-based global vector representation introduced in [36] as follows. Let v_c denote the fixed dimensional vector for a concept that we want to create. Our goal is to find vector v_c for all the concepts such that the following objective is minimized.

$$J = f(X_{ij}) \left({v_i}^T v_j + b_i + b_j - \log(X_{ij}) \right)^2. \tag{2}$$

The terms b_i and b_j denote the biases. X_{ij} denotes the number of times two concepts co-occur within a fixed length window, while $f(X_{ij})$ is a non-decreasing weighting function whose rate of change diminishes as X_{ij} increases. We would like to add that, with the advent of neural models, pre-trained language models could be a natural choice. However, the choice of our approach is primarily guided by the following two factors: i) the above approach is computationally fast, and thus it can create vectors directly from the ambient corpora, and ii) since dimension plays an important role, we can control the dimension simultaneously. This obviates further fine-tuning, which we need to perform for neural models. We provide experimental results in Section 5.4 to support our justification.

For the document nodes, we once again create another fixed dimensional vector that takes into consideration only the pedagogical concepts present in it and ignores the other words. Formally, if $C_1, C_2, \ldots C_n$ be the pedagogical concepts present in a document d, then the vector for document d is defined as

$$v(d) = \frac{1}{n} \sum_{i=1}^{n} v(C_i). \tag{3}$$

Once again, we compare the performance of our approach to that of full document representation produced by BERT in Section 5.4.

Edge properties

The graph contains three types of edges. Thus, we use different

measures to assign weights to different categories of edges as follows. The broad goal of the edge features is to allow the graph to pass information from one node to another in a unified framework, which in turn helps create better node representation in the downstream model components (Graph attention network).

Document-to-Concept. The main goal of this edge is to represent the importance of a concept in a document it appears in. This will help us get a better semantic representation of the concept nodes by focusing more on the appropriate node in the graph attention mechanism. A natural and obvious choice for such a goal is to use a vector space model or weighted probabilistic model that leverages document-specific concept saliency and collection-specific concept rareness information. However, these models fail to factor in the local distribution of the concepts [34]. Inspired by the work proposed in [34], we introduce a model based on localized extreme statistics for a given concept. The precise goal of the model is as follows. Given a document d and a concept $c \in d$, with frequency f_{dc}, we would like to determine the probability that f_{dc} lies in the extreme right tail based on the distribution of maximum values of other concepts in the documents most similar to d.

Formally, let $d_1, d_2, \ldots d_k$ be the k most similar documents to d based on their document vectors created by Equation 3. Let $X_1, X_2, \ldots X_k$ be the random variables representing the distribution of concepts in documents $d_1, d_2, \ldots d_k$ respectively. We are interested in the distribution of $X_{max} = max(X_1, X_2, \ldots X_k)$. Gumbel [17] shows that X_{max} obeys the following cumulative distribution function

$$F_e(x) = \exp^{-\exp\left(-\frac{x-\alpha}{\beta}\right)},$$

where α and β are the model parameters, which are estimated from the maximum values of $X_1, X_2, \ldots X_k$ denoted as $\mathcal{M} = \{m_1, m_2, \ldots m_k\}$. Finally, the weight between a concept c and a document d is computed by

$$e_{dc} = F_e(f_{dc}) \times IDF(c), \tag{4}$$

where $IDF(c) = \log\left(\frac{N}{df(c)}\right)$ is the inverse document frequency of the concept c. N and $df(c)$ denote the number of documents in the collection and the number of documents containing the concept, respectively. We estimate α and β using the following equations (a standard approach described in [17]).

$$\alpha + 0.58 \cdot \beta = SM,$$

$$\frac{\pi \cdot \beta}{\sqrt{(6)}} = SD,$$

where SM and SD denote the mean and standard deviation of the elements in $\mathcal{M}$.

Concept-to-Concept. The feature of this edge encodes the contextual resemblance of the two concepts represented by the edge. For this type of edge, we use *point-wise mutual information* (PMI) to obtain the edge weight between two concept vertices. We connect two vertices only if the PMI score is higher than zero and weight is the PMI value. The formula of PMI is given below:

$$\begin{aligned} pmi(c_i, c_j) &= \log \frac{p(c_i, c_j)}{p(c_i).p(c_j)}, \\ p(c_i, c_j) &= \frac{N(c_i, c_j)}{N}, \\ p(c_k) &= \frac{N(c_k)}{N}, \end{aligned} \tag{5}$$

where $N(c_i, c_j)$ is the number of sliding windows that contain both c_i and c_j, $N(c_i)$ is the number of sliding windows that only contain c_i, $N(c_j)$ refers the number of sliding windows that only contain c_j and N is the number of sliding windows in the total document corpus. This edge in our model is labeled as e_{cc}.

Document-to-Document. For document-to-document edges, the weight e_{dd} is computed as

$$e_{d_1 d_2} = \frac{< v(d_1), v(d_2) >}{||d_1|| \times ||d_2||}, \tag{6}$$

where d_1 and d_2 are two documents and $v(d)$ denotes the dense vector of d created from the concepts in d.

3.3 The Neural Model

We use the heterogeneous graph constructed above into an end-to-end neural model to predict the prerequisite relations. The neural model has two inter-connected sub-networks. The first sub-network deals with the node/concept representation leveraging the entire graph structure. On the other hand, the second sub-network uses the concept representations produced by the first sub-network, applies further non-linear transformations for better feature extraction, and finally predicts the target labels. The first and the second sub-networks are termed as the node representation network and the prediction network, respectively. We re-emphasize that the two sub-networks are trained jointly. Thus, the parameters of the graph attention network are updated based on the loss value produced in the prediction module. This ensures that the concept represented by the first sub-network is specifically engineered for the desired task and not general-purpose features.

3.3.1 Node Representation Network. The main goal of this module is to leverage the graph structure to create representations for nodes. To that end, we utilize a graph attention network on our heterogeneous graph, followed by another sub-network to identify the concept-concept prerequisite relationship. Graph Attention Network (GAT) [43] is an attention-based architecture used to perform various tasks on graph-structured data. The particular attention mechanism utilized by our Graph Attention Network closely follows the work of Bahdanau et al. [3].

The input to the GAT model is a set of node features. Our heterogeneous graph consists of two types of nodes: 1) document type and 2) concept type; the input feature of these two types of nodes is document embedding and concept embedding, respectively. We define node features, $\mathbf{v} = \{\vec{v_1}, \vec{v_2}, ..., \vec{v_N}\}$, where $\vec{v_i} \in \mathbb{R}^F$ is the initial node representation of dimension F for a concept node C_i and N represents the total number of nodes.

We transform these feature representations $\mathbf{v}$ into a new feature representation $\mathbf{v'}$ using a GAT layer. $\mathbf{v'} = \{\vec{v_1'}, \vec{v_2'}, ..., \vec{v_N'}\}$, where $\vec{v_i}' \in \mathbb{R}^{F'}$ and F' is the new feature dimension.

A shared linear transformation is used to transform input features into higher-level features, which is parameterized by a weight matrix, $\mathbf{W} \in \mathbb{R}^{F' \times F}$, and it is applied to every node. A *self-attention* mechanism has been employed on the concept pair (C_i, C_j) to compute *attention coefficient* c_{ij} between them. The attention coefficient c_{ij} indicates the importance of vertex C_j's feature to vertex C_i. The formula of the attention coefficient is as follows:

$$c_{ij} = a(\Theta\vec{v}_i, \Theta\vec{v}_j), \tag{7}$$

where $a \in \mathbb{R}^{F'} \times \mathbb{R}^{F'}$ implies a shared attentional mechanism. In our concept prerequisite identification task, if a particular concept node C_i is a prerequisite of another concept node C_j (indicates $C_i \rightarrow C_j$), then concept C_j cannot be the prerequisite of C_i. This signifies the value of attention coefficient c_{ij} is asymmetric, which implies the importance of node C_i to C_j is different from C_j to C_i. To perform *masked attention*, we feed our heterogeneous document-concept graph into the attention layer. We only obtain the c_{ij} for nodes $C_j \in \mathcal{N}_i$, where $\mathcal{N}_i$ contains all the immediate neighbor nodes of vertex C_i in the graph. To make attention coefficients comparable across different node pairs, we normalize them across all choices of vertex C_j utilizing the *softmax* activation function as follows -

$$\alpha_{ij} = \frac{exp(c_{ij})}{\sum_{k \in \mathcal{N}_i} exp(c_{ik})}. \tag{8}$$

After substituting Equation 7 in Equation 8, we get the following equation for the normalized attention coefficient.

$$\alpha_{ij} = \frac{exp\left(\sigma\left(\vec{\mathbf{a}}^T\left[\Theta\vec{v}_i \parallel \Theta\vec{v}_j\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} exp\left(\sigma\left(\vec{\mathbf{a}}^T\left[\Theta\vec{v}_i \parallel \Theta\vec{v}_k\right]\right)\right)}, \tag{9}$$

where α_{ij} is the normalized attention coefficient which is asymmetric, the attention mechanism a is a single-layer feedforward neural network, parameterized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, T denotes transposition, $\parallel$ is the concatenation operation, and σ implies the LeakyReLU nonlinearity with negative input slope $\alpha = 0.2$. If the graph has multi-dimensional or single-dimensional edge features e_{ij}, the attention coefficient α_{ij} is computed as follows:

$$\alpha_{ij} = \frac{exp\left(\sigma\left(\vec{\mathbf{a}}^T\left[\Theta\vec{v}_i \parallel \Theta\vec{v}_j \parallel \Theta_e e_{ij}\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} exp\left(\sigma\left(\vec{\mathbf{a}}^T\left[\Theta\vec{v}_i \parallel \Theta\vec{v}_k \parallel \Theta_e e_{ik}\right]\right)\right)}. \tag{10}$$

Since we have three types of edges in our heterogeneous resource-concept graph, we have employed Eq:10 to compute the attention coefficient in our implementation.

After the attention coefficients between node pairs have been normalized, these coefficients are utilized to create the final representation for every node. The following formula signifies the new feature representation $\vec{v_i'}$ for the concept C_i.

$$\vec{v_i'} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}\Theta\vec{v}_j\right), \tag{11}$$

where σ is the sigmoid activation function.

Compared to Graph convolutional Network (GCN), Graph Attention Network (GAT) implicitly assigns *different importance* scores to the nodes present in the same neighborhood, which is very effective for our concept prerequisite relation prediction task. We get a better feature representation of every node in our heterogeneous graph using this attention mechanism. For each concept pair (C_i, C_j), we feed both the hidden concept representations $(\vec{v_i'}, \vec{v_j'})$ into the prediction Network to identify whether a concept C_i is a prerequisite of concept C_j or not.

3.3.2 Prediction Network. In this section, we detail the part of the network that predicts the edge label. Specifically, given two concepts C_i and C_j as ordered pair, this module predicts the binary label $y_{ij} \in \{0, 1\}$ for the edge. Thus, if the ordered pair in the network with the feature representations $(\vec{v_i'}, \vec{v_j'})$ of the concept pairs (C_i, C_j) and the network predicts $y_{ij} = 1$, then C_i is the prerequisite concept for C_j. Otherwise, C_i is not a prerequisite concept for C_j, and C_j can still be a prerequisite concept for C_i that is determined by the label (y_{ji}) produced by the network with the input pair $(\vec{v_j'}, \vec{v_i'})$.

This sub-network takes the concept representations produced by the attention network and further fine-tunes it by applying a non-linear transformation with ReLU activation. In order to do so, each concept from the concept pair is passed through an *identical* sub-networks. *Identical* means that they have the same configuration with the same parameters and weights. Parameter update is equivalent across both the sub-networks. The main purpose of the network is to find a similarity score between two inputs by comparing their feature vectors.

Formally, we first feed the hidden representations $\vec{v_i'}$ and $\vec{v_j'}$ for the concepts C_i and C_j, respectively, into a feed-forward neural network (parameterized by W_s and b_s) followed by a ReLU nonlinearity and obtain $\vec{C}_i$ and $\vec{C}_j$.

$$\vec{C}_i = ReLU(W_s \cdot \vec{v_i'} + b_s), \tag{12}$$

$$\vec{C}_j = ReLU(W_s \cdot \vec{v_j'} + b_s), \tag{13}$$

where $\vec{v_i'}$ and $\vec{v_j'}$ are the output from the Graph Attention Network for concept C_i and concept C_j respectively. Then the prerequisite score of (C_i, C_j) is obtained as

$$p(C_i, C_j) = \sigma(W^T x + b), \tag{14}$$

where

$$x = [\vec{C}_i; \vec{C}_j; \vec{C}_i - \vec{C}_j; \vec{C}_i \otimes \vec{C}_j] \tag{15}$$

W and b are the network parameters, σ denotes the sigmoid activation function, $\otimes$ denotes hadamard product, $-$ is the subtraction operation, and $[.;.]$ implies the concatenation of vectors.

We use the binary-cross-entropy (BCE) loss as follows -

$$\mathcal{L}_{bce} = \frac{1}{|D|} \sum_{(C_i, C_j, y_{ij}) \in D} -[y_{ij} \cdot \log\big(p(C_i, C_j)\big) + (1 - y_{ij}) \cdot \log\big(1 - p(C_i, C_j)\big)], \tag{16}$$

where $\mathcal{L}_{bce}$ denotes binary-cross-entropy loss, D is the training dataset, and $y_{ij} \in \{0, 1\}$ is the gold standard of concept pair (C_i, C_j). We follow the prerequisite relation convention given in Equation 1. For the ground-truth label 0, there could be two possibilities for a concept pair (C_i, C_j): i) C_i is not a prerequisite of C_j, but C_j could still be the prerequisite for C_i and ii) C_i and C_j are unrelated. As we

Table 1: Dataset Statistics.

Dataset	|D|	|C|	$\|C_{Preq}\|$
MOOC-DSA	449	266	479
MOOC-ML	548	345	1735
Lecture Bank	897	208	921
University Course	654	411	1007

mentioned before, the first case is handled by feeding the opposite ordered pair (C_j, C_i) into the model.

4 EXPERIMENTAL SETUP

In this section, we detail the datasets used in our experiments, the baseline models (both traditional as well as state of the art), and the hyper-parameters used in our experiments. Our experiments aim to address the following key questions.

(1) How does the proposed graph neural network-based approach perform compared to the traditional supervised approaches?
(2) How does the proposed model compare against existing deep learning approaches?
(3) Since the heterogeneous graph has three different types of edges, what role does each of the edge types play in determining the prerequisite concepts?
(4) What is the impact of various components of the proposed model on its performance?

4.1 Datasets

We conducted experiments on four benchmark educational datasets from different domains. All these four datasets are publicly available and have been used as the primary benchmark in state-of-the-art models. Table 1 shows the statistics of all four datasets. In Table 1, column $|D|$ denotes the total number of documents in which the concepts are present, $|C|$ represents the total number of concepts, and $|C_{Preq}|$ is the total number of concept prerequisite relations. The descriptions of each of the datasets are given below.

- **MOOC-DSA**: This dataset is derived from Massive Online Open Course (MOOC) [1] which is used in [35]. It contains concepts from *Data Structure and Algorithms (DSA)* and consists of 266 concepts for various university data structure courses. Each course contains multiple video lectures along with video subtitles. Each subtitle implies a specific document, so the total number of documents for DSA is 449. The dataset contains 479 concept prerequisite relation pairs.
- **MOOC-ML**: This is another MOOC [2] dataset used in [35]. It contains concepts from machine learning. It has 345 concepts along with 548 documents and 1735 concept prerequisite relation pairs. Concepts and documents have been collected from various university courses.
- **Lecture Bank:** This dataset [3], used in [23], contains 897 English lecture files from 60 courses covering 5 different domains, including Natural Language Processing (NLP), Machine Learning (ML), Artificial Intelligence (AI), deep learning (DL) and information retrieval(IR). This dataset has 208 concepts and 921 concept prerequisite relation.
- **University Course:** Introduced by [27], the university course [4] dataset has 654 course descriptions from various university courses. These courses include subjects such as *algorithm Design, Computer Graphics, Graph Theory, and Neural Networks* from the domain of computer science. The total number of concepts and concept prerequisite relationships for this dataset are 411 and 1007, respectively.

4.2 Baselines

We evaluate the performance and effectiveness of our proposed model with existing state-of-the-art models. The baseline models are from the traditional machine learning models (such as SVM, Random forest, etc.) with handcrafted features, link structure-based models, and models based on Graph neural networks. Our baselines have six classification-based methods and six graph-based deep learning methods. The detailed descriptions of the baselines are given below.

Traditional Baselines:

- **Standard Classification-based models** [35]: These approaches create a set of concept features based on 1) contextual and semantic relatedness, 2) word embedding, and 3) complexity level and positions. Four traditional classification models, namely, Support Vector Machines (SVM), Logistic Regression (LR), Random Forest (RF), and Naïve Bayes (NB), are used on top of these features.
- **RefD** [24]: This baseline refers to a link-based approach that helps to measure prerequisite relations among concepts. It uses a Wikipedia link structure to compute the reference distance between two concepts, followed by thresholding the distance for the final prediction.
- **PREREQ** [37]: This is a supervised approach where the concept prerequisite relations are inferred using Pairwise Latent Dirichlet Allocation [2, 5, 33] and Siamese network.

Deep Learning Baselines:

- **GAE** [23]: GAE (Graph Auto-Encoder) is an approach based on a *graph convolutional network* (GCN) [21] with a link prediction and employs an autoencoder based loss function.
- **VGAE** [23]: Variational Graph Auto-Encoders (VGAE) is an extended version of GAE which proposed an unsupervised learning framework for unstructured data, like a graph that is used for concept prerequisite relation identification. in [23].
- **R-VGAE** [22]: The Relational-Variational Graph Auto-Encoder (R-VGAE) model was implemented by taking advantage of both relational graph convolutional network (R-GCN) and VGAE. We compared our proposed method with two versions of R-VGAE. The first one is based on term-frequency and inverse document frequency features (termed as R-VGAE(T)),

[1] http://keg.cs.tsinghua.edu.cn/jietang/software/acl17-prerequisite-relation.rar
[2] http://keg.cs.tsinghua.edu.cn/jietang/software/acl17-prerequisite-relation.rar
[3] https://github.com/Yale-LILY/LectureBank
[4] https://github.com/suderoy/PREREQ-IAAI-19/

and the second one uses dense vectors of phrases (termed as R-VGAE(P)).

- **CPRL** [18]: The Concept Prerequisite Relation Learning (CPRL) model captures concept features using Relational Graph Convolutional Network (RGCN) [40] from a heterogeneous concept learning-object graph. Thereafter it predicts the prerequisite relation between concepts using the Siamese network.

- **MHAVGAE** [48]: This model leverages Multi-headed Attention and Variational Graph Auto-Encoder (MHAVGAE) to learn the prerequisite relation between concepts. They use a combination of resource graphs and concept graphs to generate initial embedding using multi-headed attention. Subsequently, they employed a variational graph auto-encoder to reconstruct the concept graph, which in turn served as the foundation for generating initial embeddings.

Summary. The most competitive baselines to our approach are CPRL and MHAVGAE. CPRL operates on the same type of graph that we use, but they use R-GCN. Our work differs from CPRL on two main points: i) we use an attention mechanism to focus on the appropriate node in the graph, and ii) we use a new model to measure the importance of a concept in the document based on localized extreme value statistics. This model turns out to be an important edge of the graph on which the graph neural network operates. On the other hand, MHAVGAE uses multi-headed attention and graph autoencoder, and our approach uses only node attention on a heterogeneous graph.

4.3 Implementation Details

In this section, we discuss our experimental setup in detail, i.e., the training data creation and the hyper-parameter settings. At first, we build the training dataset for our proposed model in such a way that the training dataset contains the same number of samples from each class (prerequisite and non-prerequisite). Specifically, we use the following steps to create the negative samples.

For all our experiments, we train our model for 500 epochs using *binary-cross-entropy* loss. We use *Adam* optimizer with learning rate 10^{-6}. Our model consists of two Graph Attention Layers; the first layer contains 128 hidden nodes, and the second layer has 512 hidden nodes. The prediction network contains a feed-forward layer with an input dimension of 512 which outputs a 64-dimension vector.

For the concept-to-concept edge feature, which we denoted as (e_{cc}), we use a sliding window size of 10 to compute the edge weight of e_{cc}. Since, in our corpus, the most minor document contains 10 words. We set the batch size to 4 and use 5-fold cross-validation to evaluate our proposed model.

All our implementation is done by the PyTorch Geometric library using 1 NVIDIA GA100 80GB GPU. The average training duration for all datasets is 8.2 hrs. Our code is available in this link: https://anonymous.4open.science/r/CPR-GAT-0548. The README.md file in the link gives instructions to run the code.

5 EXPERIMENTAL RESULTS

In this section, we report the experimental results on the four datasets. We compare our method against traditional models that do not involve neural networks, as well as the models that use neural networks (R-GCN, Graph neural nets, etc.). Throughout the result section, we use HGAPNet (Heterogeneous Graph Attention Prerequisite Network) to denote the proposed model. We use precision (P), recall (R), and the F1 score as evaluation metrics.

5.1 Comparison to Traditional Baselines

Table 2 compares the performance of HGAPNet with six traditional baselines. The results show that the proposed model consistently and significantly outperforms all existing traditional models. For the university course dataset: i) RefD does better in terms of precision, and ii) PREREQ does better in terms of recall. For the remaining instances, the trend holds for all three evaluation metrics for all four datasets.

Among the baselines, SVM seems to be the best baseline, often outperforming the other baselines measured in terms of F1 scores. Naive Bayes is, predictably, the worst performer. The link structure-based method (RefD) does not seem to be very effective for the datasets. Similarly, the LDA with the Siamese network also performs worse than SVM.

The proposed method demonstrates balanced performance in terms of precision and recall. In contrast, most baseline models struggle to maintain a good trade-off between the two. For instance, in the University Course dataset, RefD achieves high precision but low recall, resulting in a degraded F1 score. PREREQ, on the other hand, achieves high recall but low precision on the same dataset. RF exhibits high recall but low precision for MOOC-DSA and MOOC-ML datasets while yielding high precision but low recall for the lecture bank and university course datasets. Similarly, the LR method shows high precision but low recall on the MOOC-DSA and MOOC-ML datasets.

5.2 Comparison to Deep Learning Baselines

In this section, we report and analyze the performance of the existing Deep learning models for prerequisite relation prediction. Table 3 compares the performance of six models with the proposed model. The public implementation for CPRL [18] is not available, and we implemented this from scratch following the exact hyper-parameters mentioned in the paper. Though we got results similar to as reported in the paper for MOOC-DSA, MOOC-ML, and University Course dataset, we obtained much worse performance for Lecture Bank. However, for the sake of clarity, we are reporting the results as reported in the CPRL [18] paper.

Table 3 once again shows that the proposed model is the best performer almost always. It is clearly evident that the performance differences with the existing deep models are also large. Even though CPRL performs slightly better on the Lecture Bank dataset, our proposed method performs consistently better across different datasets. On the two MOOC datasets, the proposed model achieves very high scores on all three metrics. On the two larger datasets, even though the overall performance of the proposed model is relatively worse than MOOC, the difference in performance with the existing models is equally high as in MOOC datasets.

Table 2: Performance of the proposed model (HGAPNet) compared to traditional baseline models.

	MOOC-DSA			MOOC-ML			Lecture Bank			University Course		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
SVM	0.705	0.624	0.662	0.668	0.577	0.619	**0.857**	0.692	0.766	0.796	0.635	0.707
LR	0.808	0.168	0.278	0.742	0.270	0.397	0.744	0.744	0.744	0.595	0.546	0.569
RF	0.344	0.715	0.464	0.375	0.669	0.481	0.855	0.681	0.758	0.739	0.480	0.582
NB	0.613	0.696	0.652	0.577	0.623	0.599	0.670	0.640	0.655	0.478	0.649	0.550
RefD [24]	0.920	0.252	0.396	0.784	0.188	0.303	0.666	0.228	0.339	**0.919**	0.415	0.572
PREREQ [37]	0.492	0.462	0.476	0.448	0.592	0.510	0.590	0.502	0.543	0.468	**0.916**	0.597
HGAPNet	**0.922**	**0.916**	**0.919**	**0.918**	**0.914**	**0.916**	0.842	**0.850**	**0.846**	0.884	0.880	**0.882**

Table 3: Performance of the proposed model (HGAPNet) compared to existing Deep Learning models.

	MOOC-DSA			MOOC-ML			Lecture Bank			University Course		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
GAE [23]	0.294	0.715	0.417	0.293	0.733	0.419	0.462	0.811	0.589	0.450	**0.886**	0.597
VGAE [23]	0.269	0.657	0.382	0.266	0.647	0.377	0.417	0.575	0.484	0.470	0.694	0.560
R-VGAE(T) [22]	0.677	0.455	0.544	0.674	0.465	0.550	0.817	0.518	0.634	0.806	0.562	0.662
R-VGAE(P) [22]	0.850	0.271	0.411	0.853	0.261	0.399	**0.872**	0.323	0.471	0.853	0.626	0.722
CPRL [18]	0.641	0.619	0.630	0.800	0.642	0.712	0.861	**0.858**	**0.860**	0.689	0.760	0.723
MHAVGAE [48]	0.885	0.672	0.764	0.881	0.665	0.758	0.848	0.656	0.740	0.863	0.734	0.795
HGAPNet	**0.922**	**0.916**	**0.919**	**0.918**	**0.914**	**0.916**	0.842	0.850	0.846	**0.884**	0.880	**0.882**

The results show that graph autoencoder and variational autoencoder models do not perform as well as the other baselines. The multi-headed variational graph autoencoder is the best baseline, which consistently outperforms the other strong Graph neural net baseline (CPRL) by a large margin in terms of P, R, and F1 scores. In summary, the proposed model, based on a graph attention network on a heterogeneous graph, is found to be more effective than graph autoencoder, graph R-GCN, and multi-headed graph attention network.

5.3 Effect of Edge Types

In the last two sets of experiments, we compare the performance of the proposed model by exploiting the entire weighted graph structure. The graph has three types of edges for three different types of relationships between the nodes: i) concept-to-concept, ii) document-to-concept, and ii) document-to-document. The primary objective was to improve the quality of concept node representation through a graph attention network. In this section, we seek to investigate the impact of each edge type on predicting concept prerequisite relations. To that end, we remove each edge type at a time (while keeping the other two edge types intact) from the entire graph and apply our model. Table 4 summarizes the results on all the datasets for the three different setups.

The results show that the removal of the concept-to-concept edge reduces the performance significantly on all the datasets. When this edge type is deleted, the graph neural net accumulates belief through the indirect links (document-to-document or concept-to-document). Document to concept edge also plays an important role, as its deletion reduces the performance noticeably on MOOC-DSA and University Course datasets and marginally reduces the performance on the other two datasets. Finally, document to document edge is the least important among the edges. In fact, the removal of the document-to-document relationship marginally improved the performance of the MOOC-ML dataset.

The proposed neural model uses the dense representation of two concepts to determine if one is the prerequisite of the other. Thus, the concept-to-concept edge provides a strong global signal for its representation. However, the concept-to-concept edge does not provide any information about the document-specific importance of a concept in the document it is present, and therefore, the document-to-concept edge alleviates the shortcoming. One plausible reason for the minimal impact of the document-to-document edge is that the weights assigned to the concept-to-concept edges incorporate document-specific information, indirectly representing document-to-document relationships.

5.4 Further Analysis

The proposed model has several components. In this section, we evaluate the roles of these components in the final performance of the model. Specifically, we address the following questions.

(1) How does other graph neural network architecture perform on the same graph structure for concept prerequisite relation extraction?
(2) In the prediction network, we use the concatenation of multiple elementary vector operations on the concept vectors produced by the graph attention layer. How does this compare to simple node vector concatenation in the prediction layer?
(3) What is the impact of localized extreme statistics for measuring concept importance in a document?
(4) How do the initial concept and document representation affect the performance?

Table 4: Impact of different edge types of HGAPNet. At a time, all the edges from a particular type are deleted. For example, the row with $e_{cc} = 0; e_{dc} = f_{dc}; e_{dd} = f_{dd}$ configuration means all the concept-to-concept edges are removed from the graph prior to graph attention network is applied.

	MOOC-DSA			MOOC-ML			Lecture Bank			University Course		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
HGAPNet	**0.922**	0.916	**0.919**	0.918	0.914	0.916	0.842	**0.850**	**0.846**	**0.884**	**0.880**	**0.882**
$e_{cc} = 0; e_{dc} = f_{dc}; e_{dd} = f_{dd}$	0.894	0.890	0.892	0.882	0.892	0.887	0.802	0.798	0.799	0.860	0.854	0.857
$e_{dc} = 0; e_{cc} = f_{cc}; e_{dd} = f_{dd}$	0.906	0.914	0.909	0.916	0.906	0.911	0.814	0.820	0.817	0.864	0.864	0.864
$e_{dd} = 0; e_{cc} = f_{cc}; e_{dc} = f_{dc}$	0.918	**0.918**	0.918	**0.924**	**0.932**	**0.928**	**0.844**	0.816	0.829	0.866	0.848	0.857

Table 5: Effectiveness of various components of the proposed model on Four datasets. The highest number in each column is bold-faced.

	MOOC-DSA			MOOC-ML			Lecture Bank			University Course		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
HGAPNet	**0.922**	**0.916**	**0.919**	**0.918**	0.914	**0.916**	**0.842**	**0.850**	**0.846**	**0.884**	**0.880**	**0.882**
Our graph + RGCN	0.603	0.620	0.611	0.800	0.742	0.769	0.808	0.810	0.809	0.690	0.712	0.701
GAT + concept concat	0.894	0.896	0.895	0.782	0.830	0.805	0.812	0.814	0.813	0.612	0.723	0.663
e_{dc} = TF-IDF	0.900	0.902	0.901	0.906	0.906	0.906	0.836	0.820	0.827	0.820	0.818	0.819
BERT (concept & doc)	0.900	0.910	0.905	0.912	**0.915**	0.913	0.818	0.812	0.815	0.760	0.764	0.762

In order to address the first question, we choose a relational graph convolution network (RGCN) for concept representation, which is one of the strong existing models. Moreover, since RGCN is designed for relation learning, this makes it suitable for determining prerequisite relations. RGCN is applied on the unaltered heterogeneous graph with all the nodes and edges intact, along with the same prediction layer as in the proposed approach. Table 5 presents the results on all four datasets. The results show that the Graph attention network is indeed significantly and consistently superior to RGCN.

For the second question, from the prediction network, we remove the component (Equation 15) that creates a new representation by concatenating the following three vectors: original concept vectors from GAT, their vector difference, and the element-wise product vector. Instead, we simply concatenate the concept vectors and use them for label prediction. Our objective is to investigate the role of a combination of concept representation through basic vector operations. Once again, the results show that plain concatenation is significantly worse than the former approach.

In the third question, we explore the role of a model that measures the importance of a concept in a document. For these experiments, we use the standard TF-IDF model ($count(c \in d) * idf(c)$, where c is the concept) for e_{dc} instead of a localized extreme statistics-based model. Table 5 clearly shows that TF-IDF is always worse than the extreme value-based model. The trend is consistent in terms of all three evaluation metrics.

In our last experiments, we investigated the effectiveness of BERT-based documents and concept representation. We use sentence BERT to create embedding for both the concept and the documents. Note that in our proposed model, documents are represented as the mean of the vectors of the concepts present in that document only. As it turned out, sentence BERT embedding is always poorer than the GloVe embedding and the mean concept document vector.

6 CONCLUSION

In this paper, we propose a model for predicting the prerequisite concept(s) for a given concept. The main downstream application of this research is to recommend the list of prerequisite concepts that a learner needs to learn before the learner attempts to learn the main concept. We model this problem as a heterogeneous property graph representing three types of relationships, namely, document-to-document, concept-to-concept, and document-to-concept, where the nodes and edges have properties. We then introduce an end-to-end prerequisite relation prediction model on this property graph based on a graph attention network and a simple prediction network that leverages node representations learned from the graph attention network. The model jointly trains the graph attention network and the prediction network, mutually reinforcing the parameter updates for these two sub-networks.

We conduct a set of experiments on four standard datasets and compare our method against a large number of baselines from traditional supervised machine learning models with handcrafted features and six state-of-the-art deep learning models on the graph. The results show that the proposed model consistently outperforms the existing state-of-the-art models from all categories by a significantly large margin in terms of precision, recall, and F1 measure.

We further investigate the role of graph structure and various model components (such as attention mechanism for node representation, document representation, and concept importance model) on the final performance of the model. We find that the edges representing concept-to-concept relatedness based on how often they co-occur in fixed window text are the most important signal for prerequisite relation prediction. On the other hand, document-to-document edges are the least important, and in one dataset, the removal of such edges, in fact, improved the performance of the model. We also find that graph attention has a significant impact on performance.

REFERENCES

[1] Rakesh Agrawal, Behzad Golshan, and Evangelos Papalexakis. 2015. Datadriven synthesis of study plans. *Data Insights Laboratories* (2015).

[2] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. 2008. Mixed Membership Stochastic Blockmodels. *NeurIPS* (2008).

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[4] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew M. Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. *CoRR* (2018).

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* (2003).

[6] Sahar Changuel, Nicolas Labroche, and Bernadette Bouchon-Meunier. 2015. Resources Sequencing Using Automatic Prerequisite–Outcome Annotation. *ACM Transactions on Intelligent Systems and Technology (TIST)* (2015).

[7] Hongxu Chen, Hongzhi Yin, Xiangguo Sun, Tong Chen, Bogdan Gabrys, and Katarzyna Musial. 2020. Multi-level Graph Convolutional Networks for Cross-platform Anchor Link Prediction. In *SIGKDD*.

[8] Penghe Chen, Yu Lu, Vincent W Zheng, and Yang Pian. 2018. Prerequisite-driven deep knowledge tracing. In *ICDM*.

[9] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. 2019. Multi-Label Image Recognition With Graph Convolutional Networks. In *CVPR*.

[10] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. 2020. Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting. *TIPS* (2020).

[11] Claudio Gallicchio and Alessio Micheli. 2010. Graph Echo State Networks. In *IJCNN*.

[12] Jibing Gong, Shen Wang, Jinlong Wang, Wenzheng Feng, Hao Peng, Jie Tang, and Philip S. Yu. 2020. Attentional Graph Convolutional Networks for Knowledge Concept Recommendation in MOOCs in a Heterogeneous View. In *SIGIR*.

[13] Jonathan Gordon, Linhong Zhu, Aram Galstyan, Prem Natarajan, and Gully Burns. 2016. Modeling Concept Dependencies in a Scientific Corpus. In *ACL*.

[14] Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, USA.

[15] Xinran He and David Kempe. 2014. Stability of influence maximization. In *SIGKDD*.

[16] Qirong Ho, Jacob Eisenstein, and Eric P. Xing. 2012. Document hierarchies from text and links. In *WWW*.

[17] G. S. James. 1959. *Journal of the Royal Statistical Society. Series A (General)* 122, 2 (1959), 243–245.

[18] Chenghao Jia, Yongliang Shen, Yechun Tang, Lu Sun, and Weiming Lu. 2021. Heterogeneous Graph Neural Networks for Concept Prerequisite Relation Learning in Educational Data. In *NAACL-HLT*.

[19] Istvan Jonyer, Diane J. Cook, and Lawrence B. Holder. 2001. Graph-Based Hierarchical Conceptual Clustering. *Journal of Machine Learning Research* (2001).

[20] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *CoRR* (2016).

[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[22] Irene Li, Alexander R. Fabbri, Swapnil Hingmire, and Dragomir R. Radev. 2020. R-VGAE: Relational-variational Graph Autoencoder for Unsupervised Prerequisite Chain Learning. In *COLING*.

[23] Irene Li, Alexander R. Fabbri, Robert R. Tung, and Dragomir R. Radev. 2019. What Should I Learn First: Introducing LectureBank for NLP Education and Prerequisite Chain Learning. In *AAAI*.

[24] Chen Liang, Zhaohui Wu, Wenyi Huang, and C. Lee Giles. 2015. Measuring Prerequisite Relations Among Concepts. In *EMNLP*.

[25] Chen Liang, Jianbo Ye, Shuting Wang, Bart Pursel, and C. Lee Giles. 2018. Investigating Active Learning for Concept Prerequisite Learning. In *AAAI*.

[26] Chen Liang, Jianbo Ye, Zhaohui Wu, Bart Pursel, and C. Lee Giles. 2017. Recovering Concept Prerequisite Relations from University Course Dependencies. In *AAAI*.

[27] Chen Liang, Jianbo Ye, Zhaohui Wu, Bart Pursel, and C. Lee Giles. 2017. Recovering Concept Prerequisite Relations from University Course Dependencies. In *AAAI*.

[28] Chen Liang, Jianbo Ye, Han Zhao, Bart Pursel, and C. Lee Giles. 2019. Active Learning of Strict Partial Orders: A Case Study on Concept Prerequisite Relations. In *EDM*.

[29] Hanxiao Liu, Wanli Ma, Yiming Yang, and Jaime Carbonell. 2016. Learning concept graphs from online educational data. *Journal of Artificial Intelligence Research* (2016).

[30] Weiming Lu, Yangfan Zhou, Jiale Yu, and Chenhao Jia. 2019. Concept Extraction and Prerequisite Relation Learning from Educational Data. In *AAAI*.

[31] Gori Marco, Monfardini Gabriele, and Scarselli Franco. 2005. A new model for learning in graph domains. In *IJCNN*.

[32] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent Models of Visual Attention. In *NeurIPS*.

[33] Ramesh Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. 2008. Joint latent topic models for text and citations. In *SIGKDD*.

[34] Jiaul H. Paik. 2015. A Probabilistic Model for Information Retrieval Based on Maximum Value Distribution *(SIGIR)*.

[35] Liangming Pan, Chengjiang Li, Juanzi Li, and Jie Tang. 2017. Prerequisite Relation Learning for Concepts in MOOCs. In *ACL*.

[36] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*.

[37] Sudeshna Roy, Meghana Madhyastha, Sheril Lawrence, and Vaibhav Rajan. 2019. Inferring Concept Prerequisite Relations from Online Educational Resources. In *AAAI*.

[38] Mohsen Sayyadiharikandeh, Jonathan Gordon, José Luis Ambite, and Kristina Lerman. 2019. Finding Prerequisite Relations using the Wikipedia Clickstream. In *WWW*.

[39] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE transactions on neural networks* (2009).

[40] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*.

[41] Alessandro Sperduti and Antonina Starita. 1997. Supervised neural networks for the classification of structures. *IEEE transactions on neural networks* (1997).

[42] Partha P. Talukdar and William W. Cohen. [n. d.]. Crowdsourced Comprehension: Predicting Prerequisite Structure in Wikipedia. In *NAACL-HLT*.

[43] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *ICLR* (2018).

[44] Shuting Wang and Lei Liu. 2016. Prerequisite concept maps extraction for automaticassessment. In *WWW*.

[45] Xiaolong Wang, Chengxiang Zhai, and Dan Roth. 2013. Understanding evolution of research themes: a probabilistic generative model for citations. In *SIGKDD*.

[46] Ryen W. White and Joemon M. Jose. 2004. A study of topic similarity measures. In *SIGIR*.

[47] Jia Wu, Zhibin Hong, Shirui Pan, Xingquan Zhu, Zhihua Cai, and Chengqi Zhang. 2016. Multi-graph-view subgraph mining for graph classification. *Knowledge and Information Systems* (2016).

[48] Juntao Zhang, Nanzhou Lin, Xuelong Zhang, Wei Song, Xiandi Yang, and Zhiyong Peng. 2022. Learning Concept Prerequisite Relations from Educational Data via Multi-Head Attention Variational Graph Auto-Encoders. In *WSDM*.

[49] Lisa Zhang, Zhe Kang, Xiaoxin Sun, Hong Sun, Bangzuo Zhang, and Dongbing Pu. 2021. KCRec: Knowledge-aware representation Graph Convolutional Network for Recommendation. *Knowledge-Based Systems* (2021).