AI-ASSISTED CONSTRUCTION OF EDUCATIONAL KNOWLEDGE GRAPHS

by
MEHMET CEM AYTEKIN

Submitted to the Graduate Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Doctor of Philosophy

Sabancı University
June 2024

AI-ASSISTED CONSTRUCTION OF EDUCATIONAL KNOWLEDGE GRAPHS

APPROVED BY

Prof. Dr. Yücel Saygın .....
(Dissertation Supervisor)

Prof. Dr. Şule Gündüz Öğüdücü .....

Prof. Dr. Hüsnü Yenigün .....

Assoc. Prof. Dr. Öznur Taştan .....

Assoc. Prof. Dr. Tevfik Aytekin .....

DATE OF APPROVAL: July 16, 2024

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Dr. Yücel Saygın for his guidance, support, and encouragement throughout the course of my research in Sabanci University. His expertise in the field of computer science and his dedication to his students have been instrumental in shaping my academic and professional career. During challenging times, he always approached my research from different perspectives, asked insightful questions, and provided critical feedback that helped me refine my work. Moreover, I would also like to thank Prof. Dr. Hüsnü Yenigün and Prof. Dr. Şule Gündüz Öğüdücü for their valuable help as members of my dissertation committee. Their understanding attitude, along with their thoughtful guidance, played a crucial role in the successful completion of my dissertation. Additionally, I also express my appreciation to Assoc. Prof. Öznur Taştan and Assoc. Prof. Tevfik Aytekin for taking place in my final dissertation presentation and providing valuable feedback.

I also gratefully acknowledge the support by The Scientific and Technological Research Council of Turkey (TUBITAK) through their 2244 Industrial PhD Program under the project number 118C056.

Finally, I extend my heartfelt appreciation to my family members, Osman Galip Aytekin and Selda Aytekin, for their constant support, patience, and understanding. Their love, encouragement, and words of wisdom were the driving force behind my determination and without them, this dissertation would not have been possible.

ABSTRACT

AI-ASSISTED CONSTRUCTION OF EDUCATIONAL KNOWLEDGE GRAPHS

MEHMET CEM AYTEKIN

Computer Science and Engineering
Ph.D Dissertation, June 2024

Dissertation Supervisor: Prof. Dr. Yücel Saygın

Keywords: automated prerequisite detection among concepts, building knowledge graphs, automated educational content generation, explainable AI, fine-tuning large language models, deriving inference rules from knowledge graphs.

Knowledge graphs are effective tools for organizing information. In this dissertation, we introduce a specialized graph called the Educational Knowledge Graph (EKG). This graph visualizes the concepts within a domain by circles and indicates their prerequisite relations with arrows. Such a visualization provides a comprehensive representation of the learning domain and shows students the appropriate order in which to learn these concepts. EKGs can be further enriched by textual information defining what each concept means and why it forms a prerequisite relation with the other concepts in the graph. Manual construction of EKGs is a challenging and time consuming task for three main reasons. First, it requires assigning precise definitions to each concept within the domain. Second, the domain experts need to evaluate each concept pair for identifying possible prerequisite relations. Third, the identified prerequisite relations must be justified. To address the first two challenges, we propose a methodology that combines machine learning techniques with expert knowledge. Given a domain name, our approach automatically generates specific descriptions for a predetermined number of concepts related to that domain and then assigns a prerequisite probability score to each concept pair using the generated descriptions. The high scored pairs are then asked to a human expert for validation in an iterative manner. With each round of expert feedback, the EKG in the background is updated dynamically and the final EKG is constructed once the expert decides to finish the interaction. In order to address the third challenge, we

describe a fine-tuning procedure for Large Language Models (LLMs) to teach them to identify and explain the prerequisite relations between the concepts. The results show that LLMs, fine-tuned according to our described procedure are effective models for prerequisite detection and can generate satisfactory explanations when asked to clarify the reasoning behind these relations. Finally, we present all the described methodologies in this dissertation in a web application. By using this application, we provide instructors with the ability to create their own AI-assisted EKGs and offer them to their students during their courses as supplementary learning materials.

Keywords: semantic search, prerequisite relation extraction, knowledge graph construction, large language models, fine-tuning

ÖZET

YAPAY ZEKA DESTEKLI EĞITIMSEL BILGI ŞEMALARININ OLUŞTURULMASI

MEHMET CEM AYTEKIN

Bilgisayar Bilimi ve Mühendisliği Doktora Tezi, Haziran 2024

Tez Danışmanı: Prof. Dr. Yücel Saygın

Anahtar Kelimeler: Önkoşul İlişki Çıkarımı, Önkoşul Grafikleri, Tavsiye Sistemleri, Açıklanabilir Yapay Zeka, Büyük Dil Modellerini İnce Ayarlamak, İlişkisel Verileri Sınıflandırma, Grafikler Üzerinde Çıkarım Kuralları

Bilgi şemaları, bilgiyi organize etmek için etkili araçlardır. Bu tezde, Eğitimsel Bilgi Şeması (EBŞ) adı verilen özel bir şemayı tanıtıyoruz. Bu şema bir alan içerisindeki kavramları yuvarlaklar ile görselleştirir ve bu kavramların ön koşul ilişkilerini oklarla belirtir. Böyle bir görselleştirme, öğrenilmek istenilen alanının kapsamlı bir temsilini sağlar ve öğrencilere kavramları hangi sırada öğrenemeye başlamalarının uygun olacağını gösterir. EBŞ'ler, her kavramın ne anlama geldiğini ve grafikteki diğer kavramlarla neden bir ön koşul ilişkisi oluşturduğunu tanımlayan metinsel bilgilerle daha da zenginleştirilebilir. EBŞ'lerin manuel olarak oluşturulması üç ana nedenden dolayı çok vakit alıcı ve zor bir iştir. Birinci zorluk, alan içindeki her kavramın spesifik tanımlarının oluşturulmasıdır. İkinci zorluk, ilgili alanın uzmanlarının alan içindeki her kavram çiftini tek tek değerlendirmesi ve önkoşul içerip içermediklerini belirtmesidir. Üçüncü zorluk ise, belirlenen her bir önkoşul ilişkisinin gerekçelendirilmesidir. İlk iki zorluğun üstesinden gelmek için, uzman insan bilgisi ile yapay zeka tekniklerini birleştiren bir metodoloji öneriyoruz. Metodolojimiz bize bir alan adı verildiğinde o alan ile ilgili belirli sayıda kavram ve onlara karşılık gelen açıklamaları üretiyor. Daha sonra bu üretilmiş açıklamaları kullanarak, her kavram çiftine bir önkoşul olasılık puanı atıyor. Yüksek puan alan çiftler daha sonra uzman bir insana tek tek soruluyor ve gerçekten önkoşul içerip içermedikleri teyit ettiriliyor. Her uzman geri bildiriminde arka planda eş zamanlı olarak EBŞ güncelleniyor ve uzman süreci bitirdiğinde şema oluşturulmuş oluyor. Buna ek olarak üçüncü belirtilen

zorluğun üstesinden gelmek için ise Büyük Dil Modelleri (BDM) için bir ince ayar prosedürü tanımlıyoruz. Sonuçlar, tanımladığımız prosedüre göre ince ayarlanmış BDM'lerin önkoşul tespiti için etkili modeller olduğunu ve kavramlar arası önkoşul ilişkilerini mantıklı gerekçelere dayandırarak açıkladığını gösteriyor. Son olarak, bu tezde anlatılan tüm metodolojileri bir web uygulamasında sunuyoruz. Bu uygulamayı kullanarak eğitmenlere, kendi yapay zeka destekli EBG'lerini oluşturabilme ve bunları öğrencilerine ek öğrenme materyali olarak sunabilme imkanı sağlıyoruz.

This thesis is dedicated to my family
For their endless love and support...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The field of Artificial Intelligence (AI) is constantly evolving and finding its way into many areas of our daily lives. A wide range of industries adopt AI technologies in order to improve their operations and services. Despite this broad adoption, the integration of AI in education is progressing at a slower pace compared to other sectors. This is because teaching and learning processes involve complex human interactions and emotions, making it challenging to implement AI in this area effectively. Most efforts in AI within education focus on improving knowledge presentation and optimizing the sequence of learning. To achieve this, traditional long courses are broken down into smaller knowledge units, referred to as concepts, with labeled relations between them. Among the many types of possible relations, prerequisite relation is the most studied one as identifying prerequisite relations among concepts have significant benefits to both students and instructors. First, based on the prerequisite information, students can arrange an optimal study plan for themselves where they first start learning the basic concepts and gradually move to the more advanced ones. On the other hand, instructors can use the prerequisite information to assess the students' level of knowledge within a specific course. Automatic detection of prerequisite relations among concepts in education is a challenging AI task for researchers. This is because the AI system must understand the context and meaning behind each concept and how it relates to other concepts in an unfamiliar domain. This requires a deep understanding of the educational curriculum and the ability to analyze large amounts of data. Concepts and their relations can be stored in a special data structure called Knowledge Graph (KG). This dissertation focuses on the construction of a distinct type of KG, termed as Educational Knowledge Graph (EKG) which visualizes the domain knowledge with nodes and directed edges. Nodes are the little circles, representing the key concepts in the respective domain and the directed edges are the arrows showing the identified prerequisite relations between those concepts. Instructors can build and maintain their curriculum as an EKG, together with teaching materials and learning objectives. In addition to the prerequisite relation, other types of knowledge such as the difficulty of the concepts in the domain or the reasoning behind the prerequisite relations can also

be included in the proposed EKGs. Overall, this dissertation research consists of three main parts. In the first part, we introduce a semi-supervised methodology to construct EKGs, which are designed to show all the prerequisite relations among concepts in a chosen domain. In the second part, we analyze how Large Language Models (LLMs) can be fine-tuned to help students and instructors in education especially in the context of prerequisite detection. We also demonstrate how the outputs from fine-tuned LLMs can be integrated to EKGs to improve their utility as educational tools. Finally in the third part, we present series of experiments to test the validity of our presented approaches in part one and part two. These experiments include the assessment of our prerequisite detection methodology on benchmark prerequisite datasets, testing the reliability of our constructed EKGs, measuring the correlation between prerequisite identification and student success and a comprehensive analysis of the fine-tuned LLM outputs. Before delving into the details of our proposed methodologies, we first begin by introducing terms that will be frequently encountered throughout this dissertation.

Definition 1.0.1 (Concept)**.** *A concept is defined as an abstract thought or idea. We represent concepts as data structures with two fields: a title which is the unique name of the concept that can be composed of multiple words and a description which is a textual document that includes additional details clarifying and expanding upon the title. For instance, we may create a concept with the title "Arithmetic Operations", and then define what it is by including examples from addition, subtraction or multiplication in its description.*

Definition 1.0.2 (Domain)**.** *A domain is an area that consists of many concepts. We assume that the knowledge of the domain depends on the knowledge of its concepts. For instance, if a student wants to master the Physics domain, he or she needs to be familiar with the concepts such as force, energy, motion, and electromagnetism. Frequently, knowledge of a particular concept is dependent on the knowledge of the some other concepts within the domain.*

Definition 1.0.3 (Prerequisite Relation)**.** *The prerequisite relation is established from one concept to the other. Given a pair of concept (c_i , c_j) if knowing c_i helps to understand c_j, then there is a prerequisite relation from c_i to c_j. For instance knowing the concept "Arithmetic Operations" can be helpful in understanding the other concept "Dividing Fractions".*

1.1 Importance of Prerequisite Knowledge in Education

According to the Cambridge Dictionary (2024), a prerequisite is something that must exist or happen before something else can. In the context of education, we adopt a softer definition and assume that any concept that is helpful in understanding the other concept can be regarded as a prerequisite. Online discussions between students and instructors often revolve around the question of whether a certain course or concept is a prerequisite for understanding another Quora (2024). These discussions focus on determining the proper order of course study and identifying the essential knowledge needed to understand complex subjects. Figure 1.1 provides an illustrative example of such a discussion between an instructor and a student.

Additionally, with the ever-growing knowledge on the internet, we have also witnessed rapid growth in Massive Open Online Courses platforms (MOOCs) such as Coursera, Udemy, Udacity, Khan Academy, and many more. According to a recent report, Coursera has 118 million students and provides over 10,000 different courses MOOC report (2024). Organizing the structure of the courses and arranging a learning order among educational materials require the usage of concept prerequisite information. In the past, instructors in universities provided prerequisite information for courses within certain major programs. For example, "Discrete Mathematics" was identified as a prerequisite for "Algorithms", and "Advanced Programming" was considered a prerequisite for "Compiler Design" in Sabanci University CS Curriculum (2023). However, in the era of MOOCs, manually organizing learning resources for tens of thousands of courses is not a scalable approach.

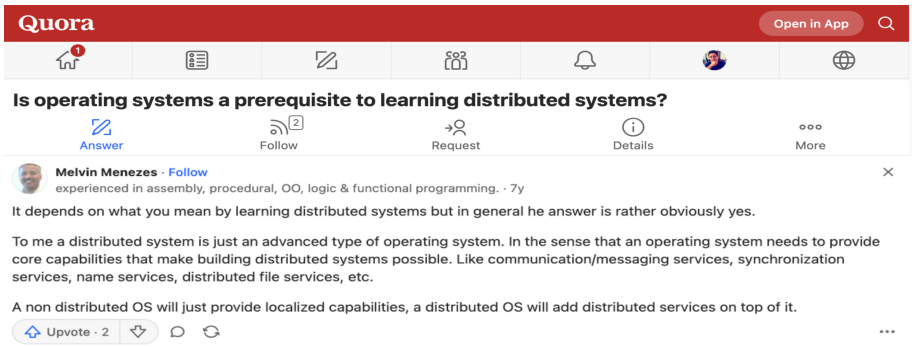

Figure 1.1 Conversation snippets from various online learning forums depicting the frequent student inquiries regarding prerequisite knowledge for certain concepts or courses. These discussions highlight the common need for prerequisite knowledge.

Since prerequisite detection plays a fundamental role in the organization and presentation of the knowledge, this dissertation focuses on identifying and explaining all the prerequisite relations within an educational domain using AI methodologies, LLMs and certain graph algorithms.

1.2 Mathematical Definition of a Prerequisite Relation

Prerequisite relation is defined as a binary relation in mathematics. Given two sets X and Y, a binary relation is a subset of their Cartesian product $X \times Y$. The Cartesian product represents the set of all ordered pairs (x, y) where $x \in X$ and $y \in Y$ and the binary relation specifies which of those pairs in $X \times Y$ should have a relation based on certain rules. For instance, in the case of "divides" relation, one can create two integer sets Z_1 and Z_2 and form the corresponding relation as a subset of $Z_1 \times Z_2$. The rules of the relation can specify that for any two integers $a \in Z_1$ and $b \in Z_2$, the pair (a, b) belongs to the "divides" relation if there exists an integer n such that $b = a \times n$. Therefore, pairs that satisfy this rule can become a part of that relation. Binary relations have three important characteristics:

1.1 **Symmetry:** A binary relation R on a set X is symmetric if for all pairs $(x, y) \in R$, the reverse (y, x) also belongs to R.

1.2 **Transitivity:** A binary relation R on a set X is transitive if whenever pairs $(x, y) \in R$ and $(y, z) \in R$ exist, then the pair (x, z) also belongs to R.

1.3 **Reflexivity:** A binary relation R on a set X is reflexive if every element x is related to itself, meaning $(x, x) \in R$ for every x in X.

A binary relation containing a symmetry characteristic is called a binary symmetric relation. Conversely, a binary relation can also be antisymmetric. The antisymmetry is defined as follows:

$$\forall a, b \in X, \text{ if } (a, b) \in R \text{ and } (b, a) \in R \text{ then } a = b$$

A binary relation R on a set X is considered antisymmetric if, for any pair of elements a and b in the set, the condition where both (a, b) and (b, a) are in relation R necessitates that a and b must be the same element. This ensures that the relation cannot have two distinct elements that are mutually related in both directions unless they are identical. This characteristic is the opposite of symmetry, where mutual relations between distinct elements are permitted. Likewise a binary relation can be also irreflexive, which is defined as:

$$\forall x \in X, (x, x) \notin R$$

This definition signifies that a relation R on a set X is irreflexive if no element in X is related to itself. That is, for every element x in the set X, the pair (x, x) does not belong to the relation R. This property is the opposite of reflexivity, where every element must relate to itself. Irreflexivity ensures that reflexive pairs are explicitly excluded from the relation. A relation that is both antisymmetric and irreflexive is called an asymmetric relation.

Prerequisite relation is an asymmetric and transitive relation. This means that if a concept pair (a, b) is an element of a prerequisite relation P that is defined on a concept set C, then concept pair (b, a) should not be in P due to the requirement of the antisymmetric property. Likewise, all the concept pairs which contain same concepts (a, a) are not the elements of P. Finally due to the requirement of the transitivity property, if (a, b) and (b, c) are the elements of P, then concept pair (a, c) is also the element of P. We also define the set P_0 which contains pairs that do not have a prerequisite relation. The union of two sets $P_0 \cup P$ corresponds to the set cartesian product $C \times C$.

The identification of prerequisite relations is treated as a binary decision problem where we determine whether a given pair (a, b) from the set $C \times C$ belongs to P or P_0. The assignment of pairs can be done manually by an expert, automatically by an AI model, or through collaboration between an expert and an AI model. Assignment of one pair effects the future assignments of the other pairs. For instance, if a pair (a, b) is assigned to set P, it should be immediately inferred that the pair (b, a) belongs to set P_0 due to the antisymmetry property of the prerequisite relation. We name the pairs in P as positive and pairs in P_0 as negative. Moreover, in order to indicate which pairs are assigned and which pairs are inferred, we use the term direct and indirect. For instance, if pairs (a, b) and (b, c) are assigned to set P, they are called direct positive pairs and the pair (b, c) becomes an indirect positive pair because it can be inferred from the direct pairs (a, b) and (b, c). Likewise, if (a, b) is a direct positive pair and (a, c) is a direct negative pair, then (b, c) becomes an indirect negative pair. We describe in detail all of those inference rules that rise from the properties of the prerequisite in Chapter 3.

1.3 Representing Prerequisite Relation Using Graphs

A binary relation on a finite set can be represented using a directed graph DAG (2024). In this dissertation, we demonstrate the prerequisite relations in a domain using EKGs. Since the prerequisite relation is asymmetric and transitive, it can be represented with a directed acyclic graph (DAG). The acyclicity of the graph is a result of the asymmetry property of the prerequisite relation. In the below, we introduce the related terminology from Graph Theory and explain the specific characteristics of EKGs.

Definition 1.3.1 (Directed Graph)**.** *A directed graph G = (N, E) consists of a set of nodes (also called vertices) N and a set of edges E that are ordered pairs of distinct nodes from N. Nodes are represented as small circles and directed edges between nodes are represented as arrows.*

In EKGs, nodes represent the concepts in the set C, and the edges represent the direct positive pairs.

Definition 1.3.2 (Simple Directed Path)**.** *A simple directed path from node A to node B in a graph is a sequence of edges that originates at node A and terminates at node B, consisting of two or more edges. This sequence allows us to reach B from A by following the directed edges, and all nodes in the path are distinct.*

In EKGs, simple directed paths between two nodes represent the indirect positive pairs.

Definition 1.3.3 (Transitive Edge)**.** *A directed edge from node A to node B is called a transitive edge, if and only if there is a simple directed path from node A to node B.*

Figure 1.2a shows transitive edges in red. Transitive edges can also be called redundant edges because they do not provide additional information regarding the prerequisite relations of the concepts.

Definition 1.3.4 (Directed Acyclic Graph (DAG))**.** *A directed acyclic graph is a directed graph with no cycles where a cycle is a sequence of edges forming a directed path starting from a node A and ending at the same node A.*

A cycle in an EKG indicates that the asymmetric property of the prerequisite relation is violated.

Definition 1.3.5 (Minimal Educational Knowledge Graph (MEKG))**.** *A graph G_{min} is called a Minimal Educational Knowledge Graph of a directed acyclic graph G if and only if G_{min} has the same nodes as G and G_{min} has all the edges of G except the transitive edges.*

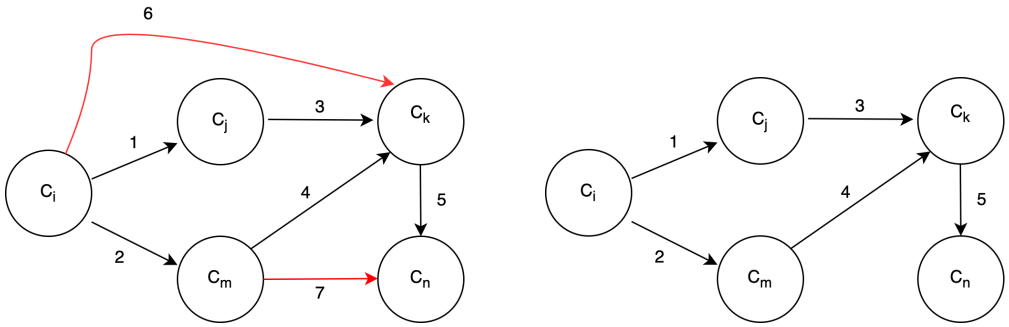

Figure 1.2 Illustration of the transformation from EKG to $MEKG$.

1.4 Prerequisite Detection Using ACE Methodology

Identifying prerequisite relations within a set of concepts C exhibits quadratic complexity, denoted as $O(n^2)$, where n is the number of concepts in C. Consequently, manual consideration of a prerequisite relation for each ordered pair in $C \times C$ becomes impractical as the size of C increases. In order to speed up this process, we propose a two-step methodology called **A**I-assisted **C**onstruction of **E**ducational KGs. In the first step, we introduce an unsupervised prerequisite scoring mechanism called Cumulative Semantic Reference (CSR) which is a function that takes an ordered concept pair (A, B) as input and returns a unique score s as output which shows how much the title of A is semantically referenced in the description of B. We assume that high values of s may be an indication of a prerequisite relation from A to B because it is likely that a concept A is essential to the understanding of concept B if A is frequently referenced or discussed in the description of B. We sort all the ordered concept pairs in $C \times C$ according to their CSR scores and store them in a list L. Pairs with scores below a predetermined threshold t are removed from L so that the list only contains pairs which are most likely to form a prerequisite relation. In the second step of ACE, we form the nodes of the EKG from the concepts of L. We then retrieve the highest scored pair (A, B) in L and ask if this pair contains a prerequisite relation to an expert. Depending on the answer of the expert, we update the EKG and move to the next highest scored pair in L. During the interaction with the expert, certain prerequisite relations can already be inferred from the previous answers of the expert, therefore, those pairs are removed from L to reduce the total number of interrogations to the expert. After the expert interaction, ACE converts the EKG into a $MEKG$ and the algorithm terminates.

CSR scores can be considered as textual evidences for the pairwise prerequisite rela-

tions. Our evaluations on student datasets from online learning platforms show that CSR scores alone can predict student performances. For instance if a concept pair (A, B) has a high score, we demonstrate that students knowing concept A perform better on the questions related to B than randomly selected students. Furthermore, experiments regarding the ACE methodology show that experts can construct accurate and reliable EKGs in a relatively short amount of time compared to the fully manual approach. The details of those experiments are reported in Chapter 5.

1.5 Explaining the Prerequisite Relations Using Large Language Models

In recent years, LLMs gained significant attention especially after the release of GPT-3, which is the largest language model as of 2023 OpenAI (2024). GPT-3 produced impressive results in natural language processing tasks (NLP) which require an understanding of language, context, and the generation of coherent, relevant text Imamguluyev (2023). One of the potential applications of LLMs is in the field of education. We already start to see collabarative works on how GPT-3 can improve the learning experience of students Kasneci, Sessler, Küchemann, Bannert, Dementieva, Fischer, Gasser, Groh, Günnemann, Hüllermeier, Krusche, Kutyniok, Michaeli, Nerdel, Pfeffer, Poquet, Sailer, Schmidt, Seidel, Stadler, Weller, Kuhn & Kasneci (2023). In order to enrich our $MEKG$s produced by our ACE methodology, we propose a special fine-tuning procedure for large language models (LLMs) that aims to teach the models to detect and explain pairwise prerequisite relations between concept pairs. Specifically, we apply this fine-tuning procedure on two different LLMs: GPT-3 OpenAI (2024) and LLAMA2 Touvron, Martin, Stone, Albert & others (2023). After the fine-tuning, LLMs learn to generate an explanatory text for each concept pair (A, B). This text outlines the reasons why A should or should not be considered a prerequisite of B. Unlike existing supervised prerequisite classifiers in the literature, this approach not only predicts prerequisite relations but also provides a rationale behind each prediction, improving the interpretability and educational utility of the models. Evaluations show that fine-tuned LLMs can achieve state-of-art performances on benchmark prerequisite datasets and the explanations they provide are similar to those made by human annotators.

1.6 Summary of Research Contributions

In summary, the purpose of this dissertation is to introduce a special type of graph data structure named $MEKG$ which is designed to represent the prerequisite relations in an educational domain with nodes and directed edges. Manual construction of such graphs is a time-consuming process. To address this challenge, the first part of this dissertation proposes a novel methodology that combines AI techniques and expert knowledge. Towards the end of the first part, the sub-relations that can be derived from the prerequisite relation are formalized. The second part of this dissertation discusses the application of LLMs in education especially in the context of prerequisite detection. This part includes the integration of additional features to $MEKG$s such as generating automated edge labels which explain the reasoning behind the prerequisite relations or improving the descriptions of the concepts using their prerequisite information. In the third and final part, we present the detailed evaluations of the discussed methodologies in the first and second part, demonstrating their effectiveness through case studies and experimental results.

Our main research contributions include:

2.1 Introduction of the AI-assisted Construction of Educational Knowledge Graphs (ACE): A novel approach that significantly speeds up the process of manual prerequisite identification by employing AI strategies.

2.2 Integration of Large Language Models (LLMs) for Prerequisite Detection: Implementation of fine-tuning techniques on LLMs, such as GPT-3 and LLAMA2, to effectively identify and provide explanations for prerequisite relations within educational content.

2.3 Assessing the correlation between the prerequisite knowledge and student success on a real world student data from an educational platform.

2.4 Presentation of our web application which can be used by instructors and students. Instructors can create their own EKGs, while students can observe the paths in the constructed EKGs to understand the prerequisite relations between different concepts within a particular domain.

1.7 Outline of the Dissertation

The rest of the dissertation is organised as follows. Chapter 2 gives related work on KGs and discusses general methodologies employed in concept-relation extraction. The chapter then presents a detailed review of the state-of-the-art methods on prerequisite detection task which is the main focus of this dissertation. Following the related work, Chapter 3 demonstrates ACE methodology which is used to identify all the prerequisite relations in a concept set. Chapter 4 introduces LLMs and their fine-tuning process for detecting and explaining prerequisite relations between concepts. Chapter 5 analyzes the effectiveness of the proposed prerequisite detection methodologies on various benchmark datasets. Moreover, this chapter also analyzes the ACE algorithm from different perspectives such as accuracy, efficiency, and scalability. Finally, Chapter 6 describes our web application which implements all the described methodologies in a user-friendly interface, enabling educators and researchers to easily create, visualize, and interact with $MEKG$s.

2. RELATED WORK

2.1 Representing Data Using Graphs

Previous research has extensively used graph data structures for representing knowledge or data in various domains. Graphs are powerful tools for organizing and visualizing complex relationships between entities, making them well-suited for representing interconnected data. For example, in the field of natural language processing, graphs have been used to model the semantic relationships between words in text documents. In bio-informatics, graphs are utilized to represent biological networks such as protein-protein interactions or metabolic pathways. Additionally, in social network analysis, graphs are commonly employed to model connections between individuals or organizations. In 2012, Google announced (KGs) Google Knowledge Graphs (2012) to structure the vast amount of information in its web documents. The usage of KG for Google showed numerous benefits such as improved search result precision Singhal (2012), enhanced data interconnectivity and the ability to answer complex queries with more accurate information Paulheim (2016). KGs offer several advantages compared to other data representation models, such as the relational model and NoSQL, mainly in terms of flexibility and scalability Hogan, Blomqvist, Cochez, d'Amato, Melo, Gutierrez, Kirrane, Gayo, Navigli, Neumaier & others (2021). In recent years, the use of KGs has been extended to various domains beyond web search, such as e-commerce, finance, or healthcare IBM Knowledge Graphs (2024). Hogan et al. discuss the application of KGs in the tourism sector to organize festivals in Chile. The graph includes entities such as cities, events, and dates, represented by nodes, and relations between these entities, represented by directed edges with different labels. KGs also enable the deduction of new knowledge that is not explicitly stored in the graph through reasoning and inference. This involves using logical rules and algorithms to derive new facts based on existing in-

formation in the graph. For instance, in the field of recommendation systems, KGs can capture the relations between users, items, and their attributes. By reasoning over this graph, new recommendations can be generated based on the preferences and behaviors of similar users or the characteristics of similar items Shokrzadeh, Feizi-Derakhshi, Balafar & Bagherzadeh Mohasefi (2023). Another example is in the domain of healthcare, where the KGs have nodes representing a patient, a disease they are diagnosed with, and the symptoms they are experiencing. By applying reasoning techniques, the graph can deduce additional information, such as potential treatments for the disease based on known effective treatments for similar cases Cui, Lu, Wang, Xu, Ma, Yu, Yu, Kan, Ling, Ho & others (2023).

Representing educational content with graphs in order to improve student learning is first proposed by Novak & Cañas (2006). Authors drew key concepts as circles and their interrelations as arrows and named their graphs as "Concept Maps". Later studies have revealed that using this means of visualization has been shown to benefit students on multiple levels, including a more satisfactory learning experience, reduced cognitive load, and better learning achievement Chiou, Tien & Lee (2015); Hirashima, Yamasaki, Fukuda & Funaoi (2015). Today the terms such as "Knowledge Graphs in Education" by Li, Cheng, Zhang, Zhu & Zhao (2023) or "Multi Source Education Knowledge Graphs" by Fettach, Ghogho & Benatallah (2022) or "Educational Knowledge Graphs" by Dang, Tang, Pang, Wang, Li & Li (2021) are more widely adopted than term "Concept Map" when referring to special graph data structures to represent the educational content. In line with the recent terminology, we use the term "Educational Knowledge Graph" (EKG) throughout this dissertation.

The initial works on EKGs assume the existence of an expert or a teacher who manually identifies the key concepts and their relations with each other in an educational domain. The manually constructed EKGs are then given to students and their learning performances are compared to the other students who study with traditional learning materials such as textbooks and lecture notes. The results showed that students utilizing EKGs generally exhibited improved retention rates and higher academic performance Novak (2010). Later on, various attempts have been made to construct EKGs either automatically or semi-automatically. For example, Aguiar et al. construct EKGs by using natural language processing techniques (NLP) on the given text document Aguiar, Cury & Zouaq (2016). The resulting graph has the extracted noun phrases as key concepts represented by nodes and verbs between them as relations represented as labeled edges. However, precision (29%) and recall (44%) were low when comparing the identified relations of the automatically constructed EKGs with EKGs constructed by experts. Lee et

al. adopted another approach which consists of utilizing burst analysis of words to establish relations between terms Lee, Park & Yoon (2015). Meanwhile, Hirashima et al. propose a semi-automatic method through which instructors and students collaboratively construct EKGs where instructors provide students with the concepts and students are asked to define the possible relations Hirashima et al. (2015). These approaches often assume an unlimited number of possible relations between concepts, making it challenging to fully automate the construction of EKGs Pinandito, Prasetya, Hayashi & Hirashima (2021). Overall, researchers face two main challenges when attempting to construct EKGs automatically. The first challenge is the extraction of key concepts in the studied domain and the second challenge is the identification of the relations between those concepts. This dissertation offers certain methodologies to automatically extract concepts from a domain however, it concentrates more on representing their prerequisite relations in EKGs after the initial concept set is established.

2.2 Concept Discovery by Keyword Extraction from Text

Named-entity recognition (NER) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc. Wikipedia contributors (2024). Entities are regarded as specific, tangible real-world objects, whereas the concepts are regarded as abstract notions or ideas. Therefore, while KGs in general, attempt to map the relationships of the real world entities in a given context, EKGs focus only on the relationships of the concepts that have educational significance. In order to discover those concepts within a given domain, researchers have applied a variety of keyword extraction algorithms. For instance, authors Chen, Lu, Zheng, Chen & Yang (2018) formulated this task as a word sequence labeling problem where the main goal is to annotate each word with a label specifying whether the word is part of a concept. The process involves assigning one of three distinct labels to each word: 1) B-CP (Begin-Concept), indicating the beginning of a concept word sequence; 2) I-CP (Inside-Concept), denoting words that continue a concept after the initial word; and 3) O (Outside-Concept), for words that are not part of a concept. In order to achieve this, authors utilize the Conditional Random Fields (CRF) model and the Gated Recurrent Unit (GRU)-based Neural Network model. The training

data for these models typically consists of text where the educational concepts or keywords are already labeled correctly. This annotated data teaches the models the patterns and relationships between words that indicate whether they form part of an educational concept. The CRF model uses the training data to learn the probabilities of different labels (like the beginning of a concept, inside a concept, or outside a concept) following each other in sequences of words. It leverages the predefined feature functions to understand how the context of certain words affects these probabilities. On the other hand, the GRU model uses training data to adjust its internal parameters, improving its ability to remember and utilize patterns seen in the data. This model learns directly from the examples provided, without needing explicit rules (feature functions), and progressively improves at predicting whether words are part of concepts based on its training. Manrique, Pereira & Mariño (2019) uses a more straightforward and generic approach to identify keywords that can be regarded as concepts. Authors first form a large corpus from a domain of interest and segment it into words, maintaining the sequence in which they appear within the original corpus. Following the segmentation, they perform a statistical analysis of words across the corpus and find meaningful unigrams, bigrams or trigrams which can be the candidates for the educational concepts. The meaningfulness of an n-gram is defined by its Pointwise Mutual Information (PMI) score which takes into account the frequency of the n-gram within the corpus and compares it to the frequencies of its individual words, essentially measuring how often words appear together compared to how often they appear independently. Apart from the PMI technique, a simple system based on noun-phrase selection is also proposed earlier for keyword extraction Barker & Cornacchia (2000). However, one major limitation of this that it may not capture the full context in which the keywords are used. Noun phrase selection primarily focuses on identifying noun phrases that are frequent within the text, but this method can miss out on relevant keywords that are not part of standard noun phrases, such as verbs, adjectives, or domain-specific terms that fall outside of the noun phrase category. Another popular method for keyword extraction is given in the work of Mihalcea & Tarau (2004) where authors present their TextRank algorithm. TextRank represents nouns and adjectives in the text as nodes and their co-occurrence as undirected edges. Each node is given a score based on its number of edges with the other nodes and high scored nodes are accepted as important words in the text. Finally if two important words co-occur in a fixed window size within the text, they are joined together. The rapid development of online glossaries and encyclopedias such as Wikipedia, led to the emergence of new concept extraction models which leveraged these vast repositories of structured knowledge. TextRazor (2024) is an example of such model which utilizes multiple external knowledge bases to detect keywords that can be the representation of the

important concepts within the text. It operates by matching the candidate keywords against Wikipedia article titles, as well as sections and chapter names of digitized books in its database. The model also uses certain machine learning techniques to measure the similarity of the context of the candidate keyword with the context of the keyword in its database to compute a relevance score. Matched keywords with high relevance scores are accepted as important concepts within that document.

2.3 Prerequisite Identification

Automatic detection of prerequisite relations among concepts in education has always been a challenging AI task for researchers Metacademy (2024). This is mainly due to the reason that an AI system is required to grasp the context and significance of each concept, and interpret its relation to other concepts within an unknown domain. The prerequisite detection task is treated as a binary classification problem in the literature. The objective of this problem is to assign concept pairs (A, B) into two classes 0 and 1 where class 1 includes pairs with a prerequisite relation from A to B, and class 0 includes pairs without such a relation. According to Talukdar & Cohen (2012), there are four cases for a concept pair (A, B);

3.1 A is a prerequisite of B

3.2 B is a prerequisite of A

3.3 A and B are related but no prerequisite relation exists

3.4 A and B are unrelated therefore no prerequisite relation exists.

In binary classification, pairs with cases 2, 3, and 4 belong to class 0 and called negative pairs while pairs with case 1 belong to class 1 and called positive pairs. In order to automate the prerequisite classification task using AI, researchers propose a wide range of models, each with its unique characteristics and constraints. A typical prerequisite classifier learns to assess certain factors about concept pairs (A, B), which can lead to the formation of a prerequisite relation from A to B. These factors are called *features*. For instance, if both A and B have dedicated articles in Wikipedia, the metadata of their articles can be used as features in prerequisite detection. For example, the AI model may learn that when the Wikipedia article categories of two concepts are different, then that pair of concepts belongs to class 0. The Maximum Entropy (**MaxEnt**) Talukdar & Cohen (2012), and the Reference Distance

(**RefD**) Liang, Wu, Huang & Giles (2015) are such models that match concepts to Wikipedia articles. To identify prerequisite relations, they utilize features such as the inter-article links, the text structure, and the categories to which the articles belong. For instance, an article about basic algebra might link to an article on calculus, suggesting that understanding algebra is a prerequisite for learning calculus. Similarly, Sayyadiharikandeh, Gordon, Ambite & Lerman (2019) uses clickstream logs of Wikipedia articles and argue that if lots of users while reading an article B go back to the article A, then concept A can be the prerequisite of concept B. Extreme Gradient Boosting (**XGBOOST**) model by Manrique et al. (2019) uses a large online Corpus (2024) which consists of 131 million English text documents. In order to identify a prerequisite relation between two concepts, authors calculate the co-occurrence statistics of those two concepts' names in the documents. The authors argue that if two concept names frequently appear together but rarely on their own, this may indicate a relationship. Moreover, if one concept consistently precedes another in related contexts, this pattern could indicate a prerequisite relation. **PREREQ** by Roy, Madhyastha, Lawrence & Rajan (2019) and **MOOC-RF** by Pan, Li, Li & Tang (2017) match concept pairs (A, B) to the online learning videos (v_A, v_B). In order to identify prerequisite relations, they count the occurrence of certain words that are semantically related to A inside the transcribed video text of v_B. They assume that high occurrence of such words indicate that concept A is a prerequisite of B. One common limitation of the models in the literature is their reliance on specific external knowledge bases, such as Wikipedia, a specific book or a course material to derive features for identifying prerequisite relationships. In the case of a Wikipedia-based approach, concept names should exist as articles so that a prerequisite relation can be identified. In the case of a corpus-based approach, concept names should be present in substantial quantities within the corpus to ensure the accurate computation of their co-occurrence statistics. Similarly, for models that use online learning materials, each concept should have its corresponding video or educational resource. There are also other approaches which extract student-related data from educational platforms to determine prerequisite relations between concepts. The typical approach is to connect the questions in the exams to various predetermined concepts and assume that if concept c_i is a prerequisite for concept c_j then a student failing on the question q_i will also fail on the question q_j. Molontay, Horváth, Bergmann, Szekrényes & Szabó (2020) propose a data-driven probabilistic student flow approach to characterize the prerequisite relations among university courses based on the success rates of students in those courses. This approach leverages large-scale student performance data to uncover hidden prerequisites among various concepts. By systematically analyzing how student successes and failures in certain courses correlate, it becomes possible to construct a probabilistic model that

predicts prerequisite structures with greater accuracy. Such models enable educational institutions to personalize learning pathways, ensuring students are guided through a curriculum that aligns with their existing knowledge and capabilities.

The described approaches in this section so far assume that the prerequisite relations between concept pairs are independent of each other. This means that when deciding a prerequisite relation from A to B for the concept pair (A, B), concept A's prerequisite relations with the other concepts do not influence the decision regarding its prerequisite relation to concept B. However, the independence assumption is not correct because all the pairs that are in prerequisite relation should reflect the mathematical properties of the prerequisite relation which are defined in Section 1.2. In the recent years, we see the emergence of graph-based learning models which represent prerequisite relations as directed graphs and assume that with enough labeled data, their models can learn to represent prerequisite relations of a set while not violating the properties of the prerequisite relation. For instance, Zhang, Lan, Yang, Zhang, Song & Peng (2022) introduced MHAVGAE, which is a multi-head attention variational graph auto-encoders model. This model takes as input an incomplete graph which consists of concepts, their resources and a set of labeled prerequisite relations and learns to complete the graph by labeling the rest of the prerequisite relations between concepts using the features from the initial graph (features from the concept resources and the existing prerequisite relations). Similarly, Jia, Shen, Tang, Sun & Lu (2021) discuss the automatic completion of a KG using the same idea. Furthermore, determining which prerequisite relations to initially label in the input graph is another research question to consider. Liang, Ye, Wang, Pursel & Giles (2018) propose an active learning paradigm in which the AI system is presented with a pool of unlabeled instances (concept pairs) and then allowed to selectively request labels for these instances based on a strategy that optimizes the learning efficiency most. The strategy chosen by the authors is uncertainty, which means that the AI model continuously learns to assign prerequisite probabilities to unlabeled instances and then requests the label of the pair whose probability is closest to 0.5 (the most uncertain). These probabilities are updated each time a concept pair is labeled. Authors show that when the initial pairs are chosen based on the uncertainty strategy, the model completes the graph more accurately compared to an alternative scenario where the initial labels are chosen from random pairs.

Although the graph based approaches show promise in identifying prerequisite relations, it is not clearly stated by the authors of the papers whether the completed graphs by the AI models actually reflect all the properties of the prerequisite relation. For instance, initially a graph can contain a prerequisite relation from concept A to B and from B to C but the AI algorithm may miss to construct an edge

from A to C which should ideally be present due to the transitivity property of the prerequisite relation. In order to overcome this problem, Liang, Ye, Zhao, Pursel & Giles (2019) introduced a check mechanism to their active learning framework. This mechanism ensured that any updates to the graph considered the transitivity, irreflexivity and the asymmetry properties of the prerequisite relation. However, the check mechanism also poses a problem. For instance, in the case where the AI-algorithm mistakenly assigns a prerequisite relation from A to B and B to C, an extra erroneous edge from A to C is also assigned. Therefore, while the check mechanism can prevent the formation of edges that violate the properties of the prerequisite relations, it can also unintentionally propagate errors, leading to a series of incorrect prerequisite relations. Consequently, we argue models that combine expert knowledge with AI capabilities should be more preferable in detecting prerequisite relations.

Yu, Wang, Zhong, Luo, Mao, Sun, Feng, Xu, Cao, Zeng, Yao, Hou, Lin, Li, Zhou, Xu, Li, Tang & Sun (2021) describe the design of the largest MOOC websites in China (MOOCCubeX) for adaptive learning. The website contains a repository consisting of around 4K courses, 230K videos, 358K exercises, 637K fine-grained concepts and over 296 million raw behavioral data of more than three million students. The main research objective of MOOCCubeX is to personalize the student learning experience and recommend the best possible learning paths to students. In order to achieve this, both student-related and content-related data are utilized. Content-related data comprises identified concepts and pairwise prerequisite relations of those concepts. Due to the massive size of the unlabeled pairs, experts manually label a small portion of them, which is then used by a neural network (NN) as training data. The trained NN then predicts the prerequisite relations of the unlabeled pairs by assigning probability scores to each of them. These probability scores are evaluated by the experts, and verified pairs are included in a separate pool. Another group of experts checks whether the prerequisite pairs in this pool form cyclic relations and, if so, removes the ones that cause cycles. This process is an example of a methodology which combines AI knowledge with human expertise in order to improve the reliability of the identified prerequisite relations.

2.4 Emergence of LLMs and their Potential in Education

In the recent years, LLMs gained significant attention especially after the release of GPT-3, which is currently the largest language model as of 2024 OpenAI (2024). The model showed impressive results in natural language processing tasks (NLP) such as question answering, named-entity recognition or natural language generation Imamguluyev (2023). One of the potential application of the model is in the field of education. We already start to see how GPT-3 can improve the learning experience of students in the work of Kasneci et al. (2023). For example, within the elementary school setting, the authors claim that LLMs like GPT-3 can improve critical thinking skills of students by generating prompts and questions that encourage them to think more deeply about the content they engage with. Extending this application to middle and high school students, LLMs can automatically generate practice exercises and quizzes that are specifically tailored to the curriculum, assisting students in their understanding of the subject matter. authors also argue that LLMs like GPT-3 can also contribute to lesson planning by helping educators construct comprehensive and inclusive lesson plans based on a provided corpus of documents. Khosravi, Denny, Moore & Stamper (2023) introduce the concept of *learnersourcing* which is a collaborative approach where students create educational content. This process offers a cognitive benefit from active involvement and builds a rich pool of learning materials. Authors claim that GPT-3 and similar LLMs can enrich these *learnersourcing* environments by generating initial content drafts, providing a basis upon which students can refine and build more complex learning resources. This partnership between human intelligence and AI in the creation, evaluation, and utilization of *learnersourced* content is anticipated to be pivotal for future educational models.

While the literature explores the advantages of LLMs in education, there is also considerable research on evaluating the outputs of these models to ensure their utility and reliability. Shneiderman (2020) discusses a human-centered approach to AI, arguing for rigorous testing and validation of AI systems before they are implemented in critical areas such as education. This perspective is crucial because even though LLMs like GPT-3 demonstrate remarkable performance across various tasks, they can still generate inaccurate, biased, or inappropriate content, which can have serious implications in educational settings. One area of focus within the evaluation of LLM outputs is the accuracy and relevance of the content generated by these models. Ziegler, Stiennon, Wu, Brown, Radford, Amodei, Christiano & Irving (2019) address the need for fine-tuning LLMs on specific domains or datasets to improve the performance on tasks that require domain-specific knowledge. This fine-tuning process must be evaluated to maintain content quality. In educational contexts, this may involve assessment by domain experts to ensure that the gen-

erated content aligns with learning objectives. Bender, Gebru, McMillan-Major & Shmitchell (2021) caution against the tendency of LLMs to replicate social biases present in their training data, necessitating active measures to detect and reduce such biases. In an educational context, biases in generated content can be damaging by providing misleading information to learners.

In this dissertation, we propose leveraging LLMs in the prerequisite detection task. Unlike existing models, the proposed approach does not require direct mapping of concepts to specific articles, text documents, or educational videos to learn the prerequisite relationships. Through their initial training with a vast amount of text data, LLMs can already generate accurate descriptions of concepts and comprehend, to some extent, what constitutes a prerequisite relation. For instance, LLMs such as GPT-3 OpenAI (2024) or Touvron et al. (2023) LLAMA2 are reported to be trained on billions of documents, including all English articles from Wikipedia, web texts, and books. Therefore, even without fine-tuning, LLMs can correctly answer half of the questions when asked whether A is a prerequisite of B, as shown in Chapter 5. To the best of our knowledge, this is the first study that uses LLMs in prerequisite detection. Another common limitation of the prerequisite classification models is their lack of explainability. Although the models classify pairs as 0 and 1, they do not provide an explanation as to why a concept is a prerequisite of another concept or not. Furthermore, when the number of the considered features increase, as in the case of deep learning models, which consider hundreds or thousands of features, interpretation of the class decisions becomes even more challenging. This is not specific to the prerequisite detection task. Any classification model that includes a large number of features will suffer from the same problem. Doshi-Velez & Kim (2017) highlight the importance of explainable AI, emphasizing that users must comprehend the rationale behind model output to trust and effectively utilize the system. As LLMs become more involved in educational processes, it will be crucial for researchers and practitioners to focus on how these models arrive at their conclusions to build student trust in automated educational tools. In the context of prerequisite detection, it becomes difficult to assess the accuracy of prerequisite relations identified by the AI models or to comprehend the potential reasons for incorrect classifications. The problem is especially evident in the first crowd-sourced prerequisite dataset initiated by Talukdar & Cohen (2012). The dataset contains many disagreements among annotators when asked if a given concept pair contains a prerequisite relation or not. Since the annotators do not provide a justification for their choices, the decision process of the annotators remains vague when evaluating these relations. As a solution, during fine-tuning process, we teach LLMs to generate a text that justifies why a certain concept is or is not considered as a prerequisite

for the other concept.

.

3. AI-Assisted Construction of Educational Knowledge Graphs

3.1 Problem Setup

We assume that there is a domain of interest together with a set of concepts with their textual descriptions from that domain. Our aim is to construct a $MEKG$ where nodes are the concepts and directed edges between nodes represent prerequisite relations. Our prerequisite pair scoring mechanism for building the graph is based on the references in the textual descriptions, which could be exact references or semantic references. A concept can have different explanations, and depending on its explanation, its prerequisites can vary. Consequently, we rely on references to concept names in those explanations to identify prerequisite relationships, yet the final decision is left to the expert. The expert may determine that, although references exist, they are insufficient to establish a prerequisite relationship. Conversely, the expert might decide that a concept is a prerequisite, despite the absence of references. Therefore, by incorporating expert feedback, we improve the reliability of our $MEKG$s in prerequisite identification. In the following subsections, we first explain the notion of semantic references as opposed to exact references, then we describe our scoring mechanism based on semantic references.

3.2 Semantic References Versus Exact References

If a learner frequently encounters some keywords in the textual description d_j of a concept c_i then this is an indication that one should know c_i before they may fully understand d_j. Determining which keywords are related to which concept in

a given textual description requires semantic analysis of the textual description. In our problem setting we have predetermined concepts where each concept has its own textual description and we need to check if there are references to other concepts in a given textual description. These references could be either *exact* or *semantic* references. An exact reference is identified when a concept is explicitly mentioned in the textual description while a semantic reference is observed when keywords with semantic connection to a concept are in the textual description. For example let c_j be *recurrent neural networks* along with its description d_j and let c_i be *chain rule*. If we encounter specific keywords such as "derivative," "product rule," or "quotient rule" within the description d_j, which are highly associated with the concept *chain rule* (c_i), we consider them as semantic references to c_i in d_j. However, if we directly encounter the concept name *chain rule* in d_j, we consider that as an exact reference to c_i. Based on the frequency and prevalence of semantic and exact references to c_i in d_j, we can decide if *chain rule* is a potential prerequisite of *recurrent neural networks*. The process of identifying exact references is straightforward. But, in order to capture the semantic similarity, we employ embeddings.

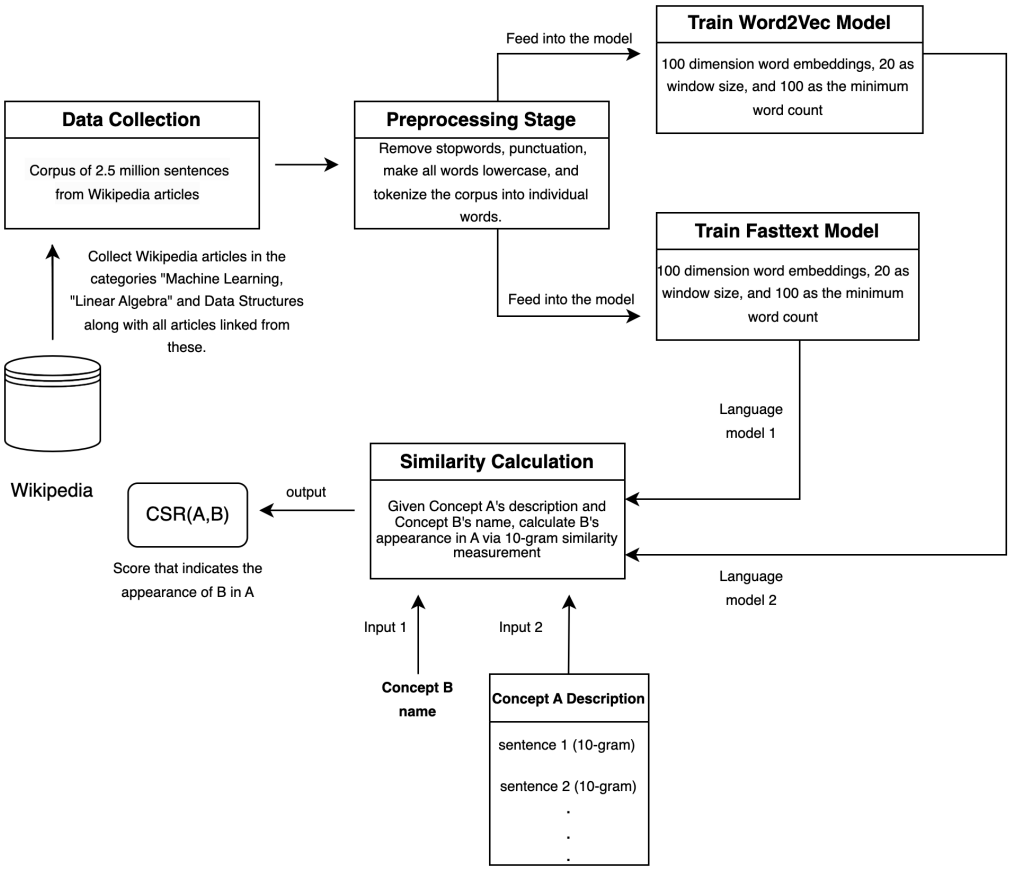

Figure 3.1 Flowchart illustrating the prerequisite detection with CSR.

3.3 Embeddings for Semantic Similarity

In order to compute semantic similarities, each word sequence must be represented by a fixed length vector called its embedding. Embeddings are learned by models through training on large amounts of textual data based on how often words appear together in the training data. By representing word sequences as fixed length vectors, the models can compare the embeddings using metrics such as cosine similarity. Cosine similarity measures the angle between two vectors, with a smaller angle indicating a higher level of similarity. Therefore, if two word sequences have similar embeddings, their cosine similarity will be high.

In this dissertation, we use 3 models that can produce embeddings with different strategies. The first two are Word2Vec and Fasttext. These models produce fixed embeddings for the words they encounter during their training phase. We calculate the similarities between sequences of words (between a 10-gram and concept name, which can be composed of any number of words). To be able to do this with Word2Vec and Fasttext, we represent two sequences by taking the average of the word embeddings for each individual word in the sequence. Since each embedding is a vector and vectors can have different lengths, we also make sure that the averaged vectors are unit vectors by dividing each vector to its L2 norm (Ecludian norm). Furthermore, Word2Vec cannot deal with out-of-vocabulary words (oov); therefore, if there is an oov in either of the sequences, we exclude it from the calculation. Fasttext, on the other hand, considers each word as a combination of character n-grams. It uses these n-gram representations to generate word embeddings. Therefore for the calculations with Fasttext, we don't check for oov because Fasttext can recognize each word even if it is oov due to its character n-gram representations.

The overall methodology for obtaining the embeddings is provided in Figure 3.1. For training Word2Vec and Fasttext models, we first formed a corpus that consists of Wikipedia articles that are of category "Machine Learning", "Linear Algebra" and "Algorithms and Data structures". We chose these domains because they align with our areas of expertise, allowing us to more accurately analyze the data. We also included the text of all the articles that are linked from those articles, eventually reaching a corpus of 2.5 million sentences. We then preprocessed our corpus by removing stopwords and punctuation marks, and by converting all words to lowercase. We tokenized the sentences into individual words and trained Word2Vec and Fasttext models on the preprocessed corpus. We used Gensim Library (2024) in Python programming language to train both Word2Vec and Fasttext models.

For Word2Vec, we set the dimension of the word embeddings to 100 and trained the model with the skip-gram algorithm with a window size of 20. We also set the minimum word count to 100, meaning that words occurring less than 100 times in the corpus are excluded from the vocabulary. We trained the Word2Vec model for 10 epochs. For Fasttext, we set the dimension of the word embeddings to 100 as in the case of Word2Vec and trained a model using skip-gram algorithm with a window size of 20. Similarly to Word2Vec, we set the minimum word count to 100. We also set the character n-gram size to range from 3 to 6, meaning that the model considers character n-grams of lengths 3, 4, 5, and 6 during training.

The third model we employed is a more sophisticated and recent model named *all-MiniLM-L6-v2*, which belongs to the category of sentence-transformers models. Sentence Transformers start by employing a pre-trained language model, such as Bidirectional Encoder Representations from Transformers (BERT) Devlin, Chang, Lee & Toutanova (2019) or Unified pre-trained Language Model (UniLM) Dong, Yang, Wang, Wei, Liu, Wang, Gao, Zhou & Hon (2019) and then train with pairs of sentences that are semantically related and non-related. The training process encourages the models to generate embeddings where semantically related sentences have close vector embeddings and non-related sentences have distant vector embeddings. The *all-MiniLM-L6-v2* employs Microsoft's MiniLM pre-trained language model Microsoft (2023) and is fine tuned on 1B sentence pairs for semantic similarity task sentence transformers (2023). The fine-tuned model can already be downloaded from Huggingface MiniLM (2023). Given a sentence with a maximum of 128 tokens, *all-MiniLM-L6-v2* produces a fixed-length sentence vector embedding of size 384. Since the model is already fine-tuned on a very large dataset for semantic similarity task, we directly use the model without any further fine-tuning and analyze its effect on our prerequisite detection methodology together with the two other models (Word2Vec and Fasttext).

3.4 Scoring Based on Semantic and Exact References

Given a set of concepts C and $c_i \in C$ where d_i is the textual description of c_i, we calculate either of the two reference scores: the Cumulative Semantic Reference score (CSR) or the Cumulative Exact Reference score (CER).

We use a sliding window strategy to split d_i into n-grams, with a fixed value of n set

to 10 and stride parameter s (which determines the distance that the window moves at each step) also set to 10. This is done to ensure a reasonable sentence size while simultaneously optimizing the speed and performance of the algorithm's execution.

Let d_i be a textual description which contains x number of 10-grams. We denote each 10-gram of d_i by s_{ij}, where $0 < j \leq x$. To detect semantic references, we consider the cosine similarities between the embeddings of concept names in C and the embedding of s_{ij}. We would like to note that the language model we use returns a single embedding for s_{ij}. CSR score of pair (c_i, c_k) is calculated as shown in Equation 3.4.

$$CSR(c_i, c_k) = \sum_j sim_{Cosine}(s_{ij}, c_k)$$

CSR scores are not symmetric, meaning that $CSR(c_i, c_k)$ may not be equal to $CSR(c_k, c_i)$ since semantic references of c_k in d_i are different than the semantic references of c_i in d_k.

Exact references are identified without any word or sentence embedding. Given a concept c_i and its description d_i with 10-grams s_{ij}, the formula for CER is:

$$CER(c_i, c_k) = \sum_j I(s_{ij}, c_k)$$

In this context, $I(s_{ij}, c_k)$ is an indicator function that outputs 1 if the 10-gram contains the concept name c_k and returns 0 otherwise.

Prerequisite ranking scores are based on either CSR or CER scores of the concept pairs. For two concepts c_i and c_k, a high $CSR(c_k, c_i)$ indicates a strong possibility that there is a direct prerequisite relation from c_i to c_k because c_i is highly referenced in the description of c_k. Although the CSR score is a good indicator for prerequisite relations, an expert interaction is still needed. This is because when both scores for $CSR(c_i, c_k)$ and $CSR(c_k, c_i)$ are high, it may cause the fully automated algorithm to make a wrong prerequisite direction assignment. Since one false positive or false negative label can effect the future labels of the other pairs in the set, we incorporate expert interaction. Therefore, in order to identify potential prerequisite pairs, we calculate the CRS scores of all ordered pairs in a concept set and sort them with respect to their scores. We then consider the top t percent of the sorted pairs as

potential direct prerequisite pairs to be evaluated by the expert.

3.5 Inference Rules

The top $t\%$ of concept pairs with the highest CSR scores are stored in a ranked list RL. The labels for these pairs are then presented to an expert for validation, starting from the first element of RL and proceeding to the last. If the expert confirms the prerequisite relation from c_i to c_j for a pair (c_i, c_j) in RL, this pair is added to the set P. Conversely, if the expert does not confirm, the pair is added to the set P_0. After each expert interaction, the EKG is updated using the two sets P and P_0. The primary goal of our methodology is to minimize the manual labeling of prerequisites by the expert. Therefore, if the label of the pair in RL can already be deduced from the previous answers of the expert, this pair is not presented to the expert and we move to the next pair in RL. Deduction of the labels is possible using inference rules on an EKG. We define 4 inference rules that are applied each time a pair (c_i, c_j) is retrieved from RL as shown in Figure 3.2.

- First rule:
  - Form the set D_j which contains node j and all of its descendant nodes.
  - Form the set A_i which contains node i and all of its ancestor nodes.
  - If there is any path from a node in D_j to A_i, then it can be automatically inferred that the label of (c_i, c_j) should be 0. This is because otherwise, an edge from node i to j would introduce a cycle in the EKG and would violate the asymmetry property of the prerequisite relation.
- Second rule:
  - Form the set D_i which contains node i and all of its descendant nodes.
  - Form the set A_j which contains node j and all of its ancestors.
  - If there is any path from a node in D_i to A_j, then it can be automatically inferred that the label of (c_i, c_j) should be 1. This is because otherwise, a missing edge from node i to j would violate the transitivity property of the prerequisite relation.

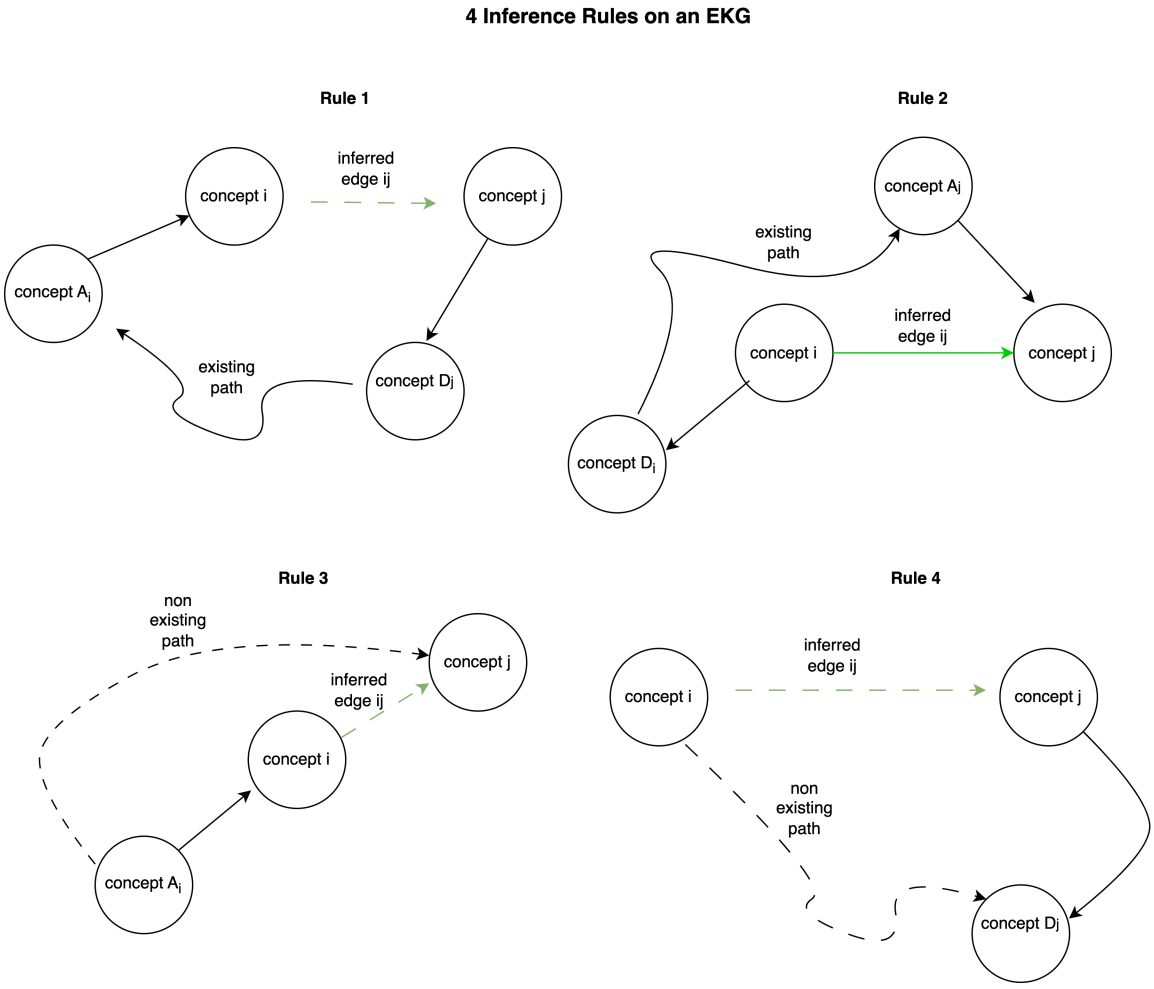

Figure 3.2 Four inference rules for a concept pair (i, j). Dashed lines represent the non existing prerequisite relations and solid lines represent the existing ones.

- Third rule:
  - Form the set A_i which contains node i and all of its ancestors.
  - For each node a in A_i, check if there is a non existing path from a to j in EKG.
  - If such non-existing path is found, then the label of (c_i, c_j) is 0. This is because if the label of (c_i, c_j) would be 1, then due to the transitivity, the expert would confirm the prerequisite relation from a to j in pair (a, j).
- Fourth rule:
  - Form the set D_j which contains node j and all of its descendant nodes.
  - For each node d in D_j, check if there is a non existing path from i to d in EKG.

– If such path is found, then the label of (c_i, c_j) is 0. This is because if label of (c_i, c_j) would be 1, then the expert wouldn't label the edge (i, d) as 0 due to the transitivity property.

3.6 ACE Main Algorithm and System Implementation

The ACE System takes a set of concepts along with their textual descriptions as input. Figure 3.4 shows a sample input set. Based on the provided input, ACE forms an $MEKG$ based on prerequisite scores, expert feedback and inference rules.Concept descriptions can be prepared by the experts or they can be obtained from existing resources. The experts also have the option to make modifications to the descriptions such as removing or adding sentences, as needed.

Algorithm 1 ACE Main Algorithm

Input:

$C = \{c_1, \ldots, c_m\}$ — the set of concepts
$d = \{d_1, \ldots, d_m \mid d_i \text{ describes } c_i \in C\}$ — textual descriptions of concepts
$P = \{\}$ — set of direct positive pairs, initially empty
$P_0 = \{\}$ — set of direct negative pairs, initially empty

Output:

MEKG for set C

1: $RL \leftarrow \text{rank}(C, d)$ ▷ Rank all the ordered pairs of C according to their CSR scores.
2: $EKG = (V, E)$, where $V = \emptyset$ and $E = \emptyset$ ▷ Initial graph with no nodes or edges.
3: **for** $k = 0$ **to** $\text{length}(RL) - 1$ **do**
4: $(c_i, c_j) = RL[k]$ ▷ Retrieve the first pair from the ranked list.
5: **if not** isInference(EKG, (c_i, c_j)) **then**
6: $answer \leftarrow$ getExpertResponse() ▷ Expert input is required.
7: **if** $answer = 0$ **then**
8: $P_0 \leftarrow P_0 \cup \{(c_i, c_j)\}$
9: Update(EKG, P_0) ▷ Update the graph knowing no prerequisite exists.
10: Reduce(EKG) ▷ Remove transitive edges.
11: **else if** $answer = 1$ **then**
12: $P \leftarrow P \cup \{(c_i, c_j)\}$
13: Update(EKG, P) ▷ Update the graph knowing a prerequisite exists.
14: Reduce(EKG) ▷ Remove transitive edges.
15: **else if** $answer = 2$ **then**
16: **break** ▷ Expert decided to stop the process.
17: **end if**
18: **end if**
19: **end for**

ACE algorithm has two distinct phases. In the first phase (line 1 of Algorithm 1), ACE assigns scores to all ordered pairs of the concept set C based on their CSR values as described in Section 3.4. This results in a list, RL, containing all concept pairs, sorted from the most likely to the least likely to form a prerequisite relation. In the second phase, ACE initializes the EKG as a null graph, where V corresponds to the set of concepts and E to the set of edges which is initially empty (line 2 of Algorithm 1). Subsequently, in (lines 3 to 15 in Algorithm 1), it iterates over concept pairs in RL. For each pair (c_i, c_j), the current graph is checked to see if the label of the pair (c_i, c_j) can be inferred automatically. If so, the function isInference, labels the pair and returns true (line 5 of Algorithm 1). Otherwise, the algorithm asks for expert feedback (line 6 in Algorithm 1), presenting the pair for evaluation. The expert's response is obtained through a graphical user interface (GUI) and

stored in the variable *answer*. Based on the expert's response, the algorithm takes appropriate actions to update the EKG. If the expert determines that c_i is not a prerequisite of c_j (line 7 in Algorithm 1), (c_i, c_j) is added to P_0 and the changes are reflected to the EKG. Alternatively, if the expert identifies c_i as a prerequisite of c_j (line 10 in Algorithm 1), pair (c_i, c_j) is included in set P and changes are again reflected to the EKG. The graph formation process may turn some edges into transitive edges. Such a scenario is shown in Figure 3.3 where red nodes in the figure indicate the current concept for which direct prerequisites are labeled. Similarly, dashed red lines indicate prerequisite candidates for labeling, whereas black edges denote actual prerequisites after the transitive edge is eliminated. Such cases are checked and newly formed transitive edges are removed by Reduce(EKG) operation (Lines 10 and 14 in Algorithm 1) to structure the graph in $MEKG$ format. The expert can also label some of the sorted pairs and retrieve the corresponding $MEKG$ without having to complete all the pairs in RL.

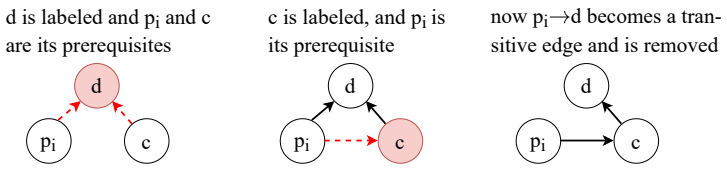

Figure 3.3 Scenario for an existing edge becoming a transitive one with newly labeled direct prerequisites.

We implemented the ACE methodology in a web application. Experts can use the system to create their own $MEKG$s by interacting with a simple GUI. The Figure 3.6 shows an instance where the expert decides that there is a prerequisite relation from *probability* to *expectation and variance* by checking the box corresponding to $C1 \rightarrow C2$. The label $C1$ corresponds to the name of the first concept; $C2$ corresponds to the name of the second concept. The CSR score is also included as a reference for the expert. Each time the expert evaluates a concept pair, she can also see the resulting $MEKG$ in the web application. Figure 3.5 shows an example $MEKG$. We can see that the DAG shows the prerequisite relations among 6 concepts. Experts may also use the web application to create their own labeled data for training supervised algorithms on a prerequisite detection task. In order to do that, ACE system has the option to convert the graph to binary labelled data in tabular form as shown in Table 3.1 where the presence of a simple directed path between a pair of concepts is represented by '1' to indicate a prerequisite relationship from the first concept to the second concept in the pair.

Concept	Concept Definition	Action
bayes_rule.txt	bayes' theorem is a mathematical formula used to calculate the probability of an event occurring, given the probability of another event that has already occurred. the theorem is named after english statistician thomas bayes (1701-1761).	Delete
conditional_distributions.txt	in probability theory and statistics, a conditional distribution is the probability distribution of a random variable given that another random variable is fixed (has occurred). more generally, a conditional distribution is the joint probability distribution of two or more random variables, given that some other random variables have already occurred.	Delete
conditional_probability.txt	conditional probability is the measure of the likelihood of an event occurring given that another event has already occurred. the concept is often used in statistical analysis to predict the probability of a certain outcome given a certain set of circumstances. for example, if it is known that someone has a 60% chance of winning a game, the conditional probability of them winning two games in a row can be calculated as 36%.	Delete
expectation_and_variance.txt	given a random variable, we often compute the expectation and variance, two important summary statistics. the expectation describes the average value and the variance describes the spread (amount of variability) around the expectation	Delete
maximum_a_posteriori_estimation.txt	in bayesian statistics, a maximum a posteriori probability (map) estimate is an estimate of an unknown quantity, that equals the mode of the posterior distribution. the map can be used to obtain a point estimate of an unobserved quantity on the basis of empirical data. it is closely related to the method of maximum likelihood (ml) estimation, but employs an augmented optimization objective which incorporates a prior distribution (that quantifies the additional information available through prior knowledge of a related event) over the quantity one wants to estimate. map estimation can therefore be seen as a regularization of maximum likelihood estimation.	Delete
probability.txt	probability is the branch of mathematics that deals with the analysis of random phenomena. the central idea of probability is that given some conditions, certain events are more likely to occur than others. probability can be used to model situations in which there is uncertainty about whether an event will occur.	Delete

Figure 3.4 A sample concept set for constructing an EKG.

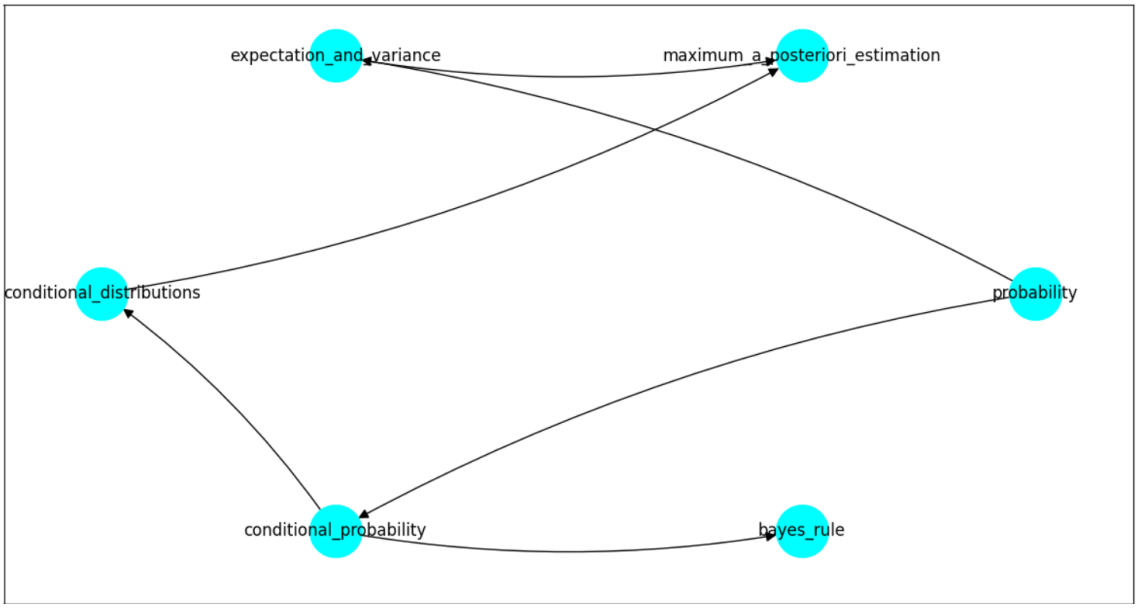

Figure 3.5 Example of a constructed $MEKG$ for 6 concepts.

3.7 Role of the Expert

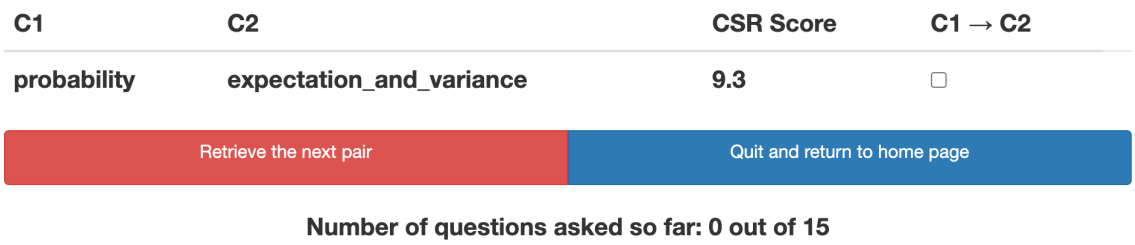

Figure 3.6 Example of an instance where the expert makes a selection from the GUI

Concept 1	Concept 2	Label
C1	C2	1
C1	C3	0
C1	C4	0
C4	C1	1
C4	C2	1
C4	C3	0

Table 3.1 Binary labeled data where 1 indicates a prerequisite relation from the first concept to the second concept and 0 means not a prerequisite

The involvement of an expert helps ensure the $MEKG$ does not contain invalid edges, assuming the expert accurately assigns direct prerequisite relationships. Note that, invalid edges in the graph can have a cascading effect, resulting in the formation of many invalid paths. Furthermore, references in a textual description indicate a potential prerequisite relation but, sometimes an expert assistance is needed in order to determine the direction of the relation because both descriptions can refer to each other with similar CSR scores. For example, in our evaluation study we have encountered the concept *linked list* whose textual description refers to concept *tree* as "*linked list* can be used to implement several other data structures such as *stack* and *tree*" and at the same time, description of concept *tree* also refers to concept *linked list* as "*trees* are implemented with *linked lists*" thus making $CSR(linked\,list, tree)$ and $CSR(tree, linked\,list)$ high and close to each other. Therefore, expert judgment becomes crucial in identifying the correct prerequisite direction between such closely connected concepts.

3.7.1 Maximum Number of Edges in MEKGs

The $MEKG$s produced by ACE algorithm are DAGs with no transitive edges. We investigate the maximum number of edges an $MEKG$ can contain. This is important

in identifying the minimum number of questions that can be asked to the expert in $MEKG$ construction. A $MEKG$ can contain maximum of $n^2/4$ number of edges. We derive this formula using Mantel's Theorem as follows:

Consider two graphs $G(N, E)$ and $G_{dir}(N, E^*)$. The graph G is a simple undirected graph, meaning that it has no edges connecting a node to itself and no multiple edges between any pair of its nodes. The graph G_{dir} is the directed version of G, in which all edges E^* are assigned a random direction, either from the first node to the second or from the second to the first.

Proposition 1. *If G_{dir} does not contain a transitive edge then G is triangle free.*

Proposition 2. *If G is a triangle free graph then we can always form a G_{dir} which contains neither a cycle nor a transitive edge.*

Since a $MEKG$ neither contains a cycle nor a transitive edge, we conclude that there will always be a G_{dir} in $MEKG$ format if G is known to be triangle free. We also know that the number of edges in G and G_{dir} are same because direction assignments do not change the edge count. Therefore, in order to find an upper bound on the number of edges a $MEKG$ can contain, we find the maximum number of edges a triangle-free graph G can contain using Mantel's Theorem.

[1]

The upper bound on the number of edges in $MEKG$ suggests that given a concept set C with n number of concepts, there can be at most $n^2/4$ number of concept pairs (A, B) which contain a direct prerequisite relation (or directed edge in $MEKG$) from A to B if n is even number. If n is odd, the formula changes to $(n^2-1)/4$. Similarly, the total number of direct and indirect relations can be at most :

$$\frac{n \times (n+1)}{2} - n$$

This is because if a pair (A, B) in $C \times C$ has a prerequisite relation from A to B, then the pair (B, A) does not have a prerequisite relation from B to A, due to the asymmetry property and due to the irreflexivity, n number of same pairs (A, A) cannot have a prerequisite relation from A to A, and the total number of pairs that can potentially hold a prerequisite relation becomes the total number of unordered pairs minus the reflexive pairs in set C.

[1] Please see Appendix for the proofs of theorems introduced hereafter.

3.7.2 Bounds On the Number of Queries to the Expert

The prerequisite scoring algorithm in ACE ranks all the ordered pairs (A, B) according to their likelihood of containing a prerequisite relation from A to B and then brings top scored pairs to the expert for consideration. If the ranking algorithm is perfect, thereby achieving 100% sensitivity (true positive rate), the top %25 pairs will include all the direct prerequisites. This is because for a concept set with n concepts, the number of total ordered concept pairs is n^2 and the maximum number of direct prerequisites is $n^2/4$ making

$$\frac{\left(\frac{n^2}{4}\right)}{n^2} = \frac{n^2}{4} \cdot \frac{1}{n^2} = \frac{1}{4} = 25\%$$

Consequently, from those direct prerequisite pairs, all the prerequisite relations will be revealed and visualized in the respective $MEKG$. If the prerequisite scoring algorithm does not make a distinction between an indirect and direct prerequisite pair but achieves 100% sensitivity (i.e every pair it brings to the expert is either a direct or indirect prerequisite pair) then it has to make sure that the expert validates $\frac{n\times(n+1)}{2} - n$ pairs. This means that in worst case, expert has to consider approximately half of the pairs to build the complete $MEKG$ assuming that the prerequisite scoring algorithm is perfect.

We call a $MEKG$ with all the true prerequisite relations as *a complete* $MEKG$. However, in practice no algorithm has a 100% sensitivity and it may be the case that the algorithm brings false positives instead of true positives to the expert. Presence of the expert guarantees that the those false positives are labeled as no prerequisites therefore, no erroneous edge is drawn in the $MEKG$. The more pairs are brought to expert, the more it is likely that all the true positives will be captured. Trivially, if a prerequisite ranking algorithm brings all the n^2 pairs to the expert, the $MEKG$ will be complete. However, this means that expert has to evaluate all the pairs for prerequisite relation which increases the manual effort a lot. Conversely, if the algorithm only brings the small portion of the pairs, then the $MEKG$ will be incomplete as lots of true positives will not be labeled by the expert. In order to demonstrate the balance between the reduction in expert effort and the completeness of the $MEKG$, we introduce two parameters t and path recall. Parameter t shows what percentage of the most likely prerequisite pairs will be evaluated by the expert and path recall shows how much of the true prerequisites are captured by the resulting $MEKG$ after the expert interaction. As described in the ACE Algorithm (see Algorithm 1), once the $MEKG$ starts being constructed from the answers of

the expert, some of the top-scored pairs' labels can be inferred and not asked to the expert. Therefore, the total reduction in expert effort for a concept set with size n becomes

$$\frac{100-t}{100}+\frac{f}{n^2}$$

where f represents the number of inferred labels. For instance, given a set C with 10 concepts, suppose $t=40$. This means that in worst case, 40 of the highest scored pairs from a total of 100 pairs should be evaluated by the expert. During the construction of the $MEKG$, if 5 labels are inferred and not asked to expert, the total number of pairs not considered by the expert becomes $60+5=65$, and the relative reduction in expert effort becomes 65%. In evaluation, we plot path recall against t to demonstrate the balance between algorithm accuracy and manual effort. Continuing from the previous example, if for $t=40$, the path recall value is 80, it means that with a 65% relative reduction in expert effort, the constructed $MEKG$ covers 80% of the total true prerequisite relations.

3.7.3 Representing Strict Partial Ordered Sets Using MEKGs

In mathematics, a strict partial order is defined as a binary relation that is irreflexive and asymmetric. Therefore, prerequisite relation can also be called as a strict partial order. The word partial means that not all the elements of the relation are comparable. On the contrary, a strict total order is an irreflexive and asymmetric relation in which all the elements are comparable. For instance, the binary relation "greater than" on a set of natural numbers N is a strict total order because for each chosen ordered pair (A,B) there is either a relation from A to B or B to A. The word strict refers to the irreflexive property of the relation and if the relation is a partial or a total order with reflexive property such as the relation "greater than or equal to", than the word strict is removed. In $MEKG$s with prerequisite relations, a prerequisite scoring algorithm in ACE helps the expert to build the graph by eliminating pairs that are not likely to have direct prerequisite relations. Similarly, for other types of relations that are strict partial order, a similar ranking algorithm can be built and the rest of the steps in ACE can be directly utilized to produce the $MEKG$ of the corresponding relation.

A subset of a strict partial order set can be a total order set. In general, the term

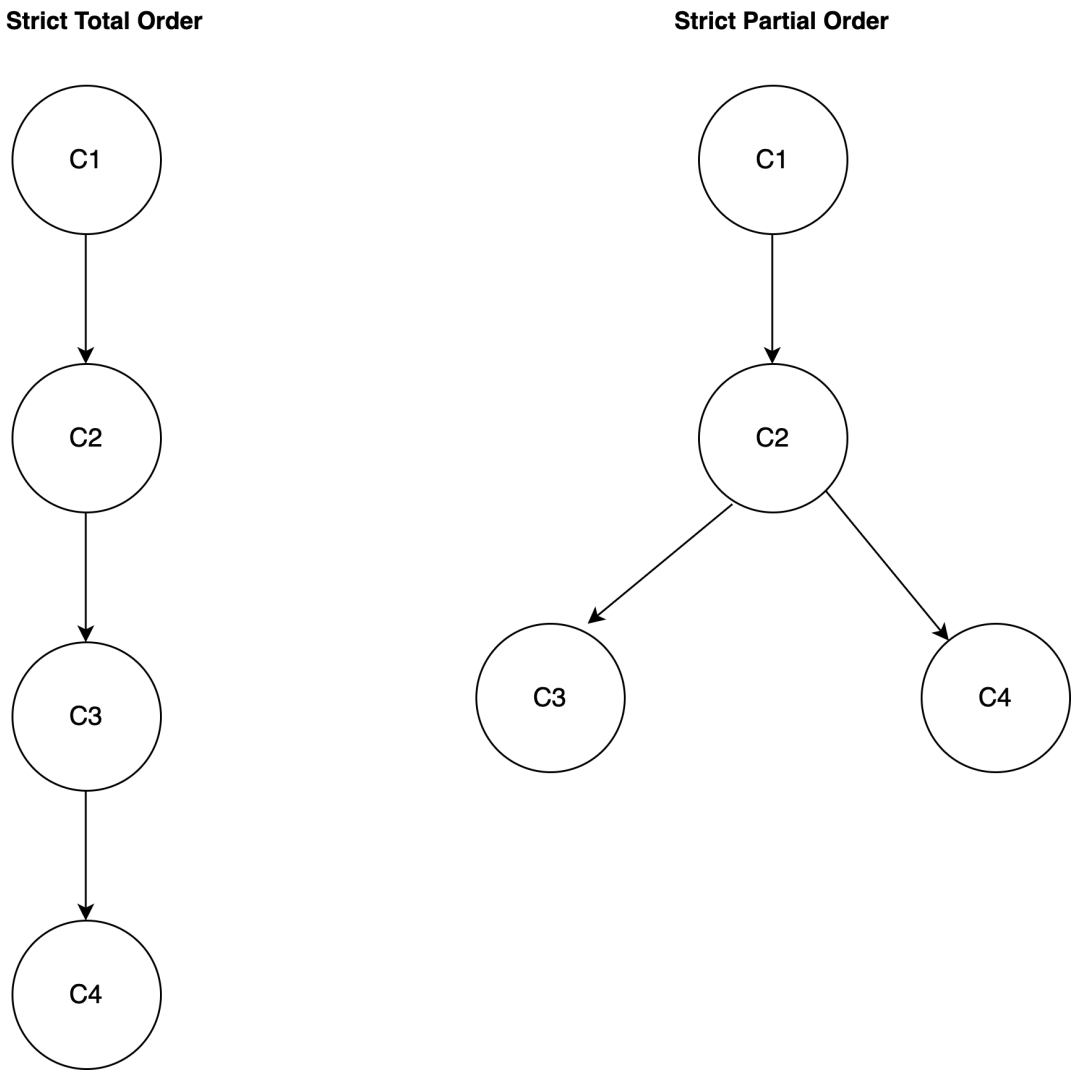

Figure 3.7 Comparison between a Partial Order and a Total Order Set.

chain is used to describe these sets. In $MEKG$s, chains correspond to the nodes of the simple directed paths in the graph. Since indirect prerequisite relations between concept pairs are simple directed paths, if A is an indirect prerequisite of B, all the nodes of path $(A \to n_1 \to n_2 \to \cdots \to B)$ form a total order set. Identification of chains in $MEKG$s allows us to define more strict relations between the ordered pairs of the chain such as "precedence" or "difficulty". Given a chain H from a $MEKG$ with prerequisite relation, the relation R on H contains pairs either (A, B) or (B, A). This means that if the length of H is n, there are exactly $n * (n-1)/2$ pairs containing the relation R. When R is defined as the difficulty relation, each pair formed from H can tell which of the concept in the pair is less difficult than the other. We can also identify the maximum or the minimum element of H. The minimum element of the chain is defined as the node which has no incoming edge and it can be interpreted as the concept whose description is the easiest to understand. Figure 3.7 shows the difference between a partial order and total order. In left, we

observe that all concepts are comparable and in right, we see that $C3$ and $C4$ are not comparable.

3.8 An Automated Algorithm to Sort Concepts based on their Difficulty

In this section, we introduce an algorithm called Difficulty Orderer (DO) which orders concepts based on their difficulty scores from easiest to hardest. The difficulty scores are determined according to the difficulty relation R which is defined as the subset of the prerequisite relation as explained in Section 3.7.3, and we assume it is a strict total order.

Given a concept pair (i, j) from a concept set C, if $(i, j) \in R$ then $(j, i) \notin R$ and i is a less difficult concept than j. Conversely, if $(j, i) \in R$ then $(i, j) \notin R$ and j is a less difficult concept than i. Moreover, since R is a strict total order, either $(i, j) \in R$ or $(j, i) \in R$.

In order to determine if concept pair (i, j) or (j, i) belongs to R, we define a comparator function F which takes a concept pair (i, j) and returns 1 if $(i, j) \in R$ and -1 if $(j, i) \in R$. The function uses the incoming Cumulative Semantic Reference (inCSR) and outgoing Cumulative Semantic Reference (outCSR) scores of the concepts. Given a concept set C, inCSR is defined as follows:

$$\text{inCSR}(i) = \sum_{x \in C \setminus \{i\}} \text{CSR}(x, i)$$

Intuitively, if a concept i has high inCSR score, this means that i is frequently referenced in the other concepts' descriptions.

Similarly, outCSR is defined as follows:

$$\text{outCSR}(i) = \sum_{x \in C \setminus \{i\}} \text{CSR}(i, x)$$

A high outCSR of a concept i indicates that i frequently references the other concepts in its description.

Difference between the inCSR and outCSR of a concept determines its difficulty. A basic concept is assumed to have a simple definition that does not involve the usage of other concepts hence a low outCSR value. Conversely, this basic concept is expected to be present in the definitions of the other concepts since it serves as a foundational knowledge element, thus leading to a high inCSR value. Therefore, the difference between the inCSR and outCSR values (inCSR - outCSR) can indicate the relative simplicity of a concept within a given concept set. Higher positive values suggest the concept is more basic and foundational, while lower or negative values might indicate that the concept is more complex or specialized, relying heavily on other concepts for its definition.

Finally we define the comparator function $F(i, j)$ as :

$$F(i,j) = \begin{cases} 1 & \text{if } \mathrm{inCSR}(i) - \mathrm{outCSR}(i) - (\mathrm{inCSR}(j) - \mathrm{outCSR}(j)) > 0, \\ -1 & \text{otherwise.} \end{cases}$$

We illustrate all the described steps in Algorithm 2. The *DO* algorithm utilizes the merge sort technique, specifically adapted to sort concepts by their difficulty levels as defined by the comparator function F. This structured ordering is useful for organizing educational content, structuring lessons, or designing curricula, effectively organizing concepts from the simplest to the most complex based on their difficulty relations.

Algorithm 2 DO Algorithm with Recursive Merge Sort

Input:

$C = \{c_1, \ldots, c_n\}$ — List of elements to sort

Output:

Sorted list of concepts from easiest to hardest based on difficulty

function DO(C)
  if length(C) > 1 **then**
    $mid \leftarrow \text{length}(C)//2$ ▷ Find the middle index
    $L \leftarrow C[1 : mid]$ ▷ Divide the list into left half
    $R \leftarrow C[mid + 1 :]$ ▷ Divide the list into right half
    $L \leftarrow \text{DO}(L)$ ▷ Recursively sort the left half
    $R \leftarrow \text{DO}(R)$ ▷ Recursively sort the right half
    $C \leftarrow \text{MERGE}(L, R)$ ▷ Merge the sorted halves
  end if
  return C
end function
function MERGE(L, R)
  $i \leftarrow 1$
  $j \leftarrow 1$
  $merged \leftarrow []$
  while $i \leq \text{length}(L)$ and $j \leq \text{length}(R)$ **do**
    if $\text{F}(L[i], R[j]) == 1$ **then**
      Append $L[i]$ to $merged$ ▷ $L[i]$ is less difficult than $R[j]$
      $i \leftarrow i + 1$
    else
      Append $R[j]$ to $merged$ ▷ $R[j]$ is less difficult than $L[i]$
      $j \leftarrow j + 1$
    end if
  end while
  while $i \leq \text{length}(L)$ **do**
    Append $L[i]$ to $merged$ ▷ Copy remaining concepts from L
    $i \leftarrow i + 1$
  end while
  while $j \leq \text{length}(R)$ **do**
    Append $R[j]$ to $merged$ ▷ Copy remaining concepts from R
    $j \leftarrow j + 1$
  end while
  return $merged$ ▷ Return merged list with concepts sorted by difficulty
end function

3.9 Complexity Analysis of CSR Computation and DO Algorithm

In order to calculate the CSR score of a concept pair (A, B), our methodology utilizes the description of A, the title of B and a language model L. As described in Section 3.4, the textual description of A is converted into a set of 10-grams and each 10-gram is vectorized by L. After that, L also vectorizes the title of B and then calculates the cosine similarity between the vectors of 10-grams of A and the vector of the title of B. The vectorization process of an n-gram depends on the chosen L. For instance, if L is trained to produce fixed word vectors (embeddings), it may use a hash data structure and a look up table to find the corresponding vectors of 10 words and average them to find the embedding of the given 10-gram which takes constant $O(10)$ time. However, if the word vectors are dynamic (i.e the embedding of each word depends on the embeddings of the previous words in the n-gram) then the process may exhibit a complexity that is up to $O(n)$ time. When a 10-gram is vectorized, its cosine similarity with the vector of title B is computed. Cosine similarity operation takes $O(s)$ time if both vectors have length s. This is because each operation (multiplying vector components, adding them for the dot product, computing the squares of components, summing them for the norms, and dividing the dot product by the product of the norms) relates directly to the number of elements in the vectors. Given that the size of the vectors and the length of the n-grams may vary depending on the chosen language model L, we treat the similarity calculation between the 10-gram of A and the title of B as having a unit cost, denoted by c. This assumption simplifies our calculations by considering these operations as constant time, irrespective of the actual vector dimensions or n-gram length.

In each concept description, if we have k number of sentences on average, then calculating $CSR(A, B)$ requires k similarity operations which has a cost of $c \times k$. Each inCSR and outCSR computation requires n number of CSR operations. Moreover, $F(i, j)$ requires 2 inCSR and 2 outCSR computations, which at total constitutes a $4 \times n \times c \times k$ cost. Since merge-sort algorithm does $n \times log(n)$ comparisons in every case (worst, best or average) for n number of items, we use $F(i, j)$ $n \times log(n)$ times. Finally, we decide that the cost of sorting n concepts based on their difficulty has a cost of $n \times log(n) \times 4 \times n \times c \times k$ which can also be expressed as having a $n^2 \times log(n)$ time complexity.

4. Discovering Prerequisite Relations Using LLMs

4.1 Introduction

This section illustrates the process of fine-tuning large language models (LLMs) to predict prerequisite relations between concepts. In the fine-tuning phase, a LLM receives a prompt asking whether one concept is a prerequisite for another. It then learns to respond in a structured format. The response includes: 1) a "yes" or "no" to indicate whether a prerequisite relationship exists, and 2) a rationale, provided in a subsequent sentence, explaining the reason for its decision. This fine-tuning strategy is then generalized to other potential classification problems in education such as student profiling or automatic assignment evaluation. The evaluation results show that the fine-tuned LLMs learn the classification tasks with ease and their answers to the test prompts are similar to those answered by human annotators.

4.2 The Study and the Method

Our methodology exploits LLMs as binary classifiers for prerequisite detection. LLMs are able to generate an output text (called *completion*) based on a given input (called *prompt*). By leveraging this capability, we introduce a specialized fine-tuning strategy where LLMs receive a concept pair (A, B), and a label l, through which they learn to generate a *completion* which states why A is a prerequisite of B if $l = 1$ and why A is not a prerequisite of B if $l = 0$. Therefore, after fine-tuning, LLMs have two main functions. First, they are able to classify any given pair to a class 0 or 1. Second, they are able to justify their assignments with *completion* text.

The second functionality is particularly important in educational applications where students need to understand why certain concepts are prerequisites for others.

4.3 Fine-tuning Process

The benchmark datasets, on which we fine-tune LLMs, consist of two separate files: a training file and a test file. The training file contains a set of concept pairs and their respective labels (0 or 1) and the test file contains another set of concept pairs whose labels should be predicted by the AI model. During the fine-tuning process, we convert each concept pair (A, B) in the training file into a question prompt as "Is concept A a prerequisite of B?". This is the question that the LLMs should learn to answer when predicting the class of a concept pair in the test file. Since we also want to teach the models how to specifically answer a question, we create two types of answers; positive and negative represented by completion string1 and completion string2 respectively as can be seen in Figure 4.1 and Figure 4.2. Answers for positive pairs (those with a class label 1) represented with completion string1 start with the keyword "Yes", followed by a special marker ";". The keyword corresponds to the correct class of the pair. Since we also want to have a justification text on why A is a prerequisite of B, we provide an additional prompt as input to LLM, which is in the form of "Why is concept A a prerequisite of B?" The answer returned from LLM is appended to completion string1 following the special marker ";". The construction of the completion string1 for a positive pair (Expected Value, Kalman Filter) is depicted in Figure 4.1. Similarly, for the negative pairs (pairs with class label 0), the answer formatted as completion string2 starts with the keyword "No", followed by the special marker ";". This is followed by a justification response from LLMs to the question "Why is concept A not a prerequisite of B?" and appended after the special marker. The construction of the completion string2 for an example negative pair (Integral, Predicate Logic) is also depicted in Figure 4.2. Upon successful fine-tuning, LLMs can determine whether one concept is a prerequisite for another and justify their decision with a generated explanation. We made the fine-tuning data publicly available so that any researcher working on the prerequisite detection problem can reuse it to fine-tune their own LLM models. Our fine-tuning procedure can also a guide for other models which work on different types of binary classification problems.

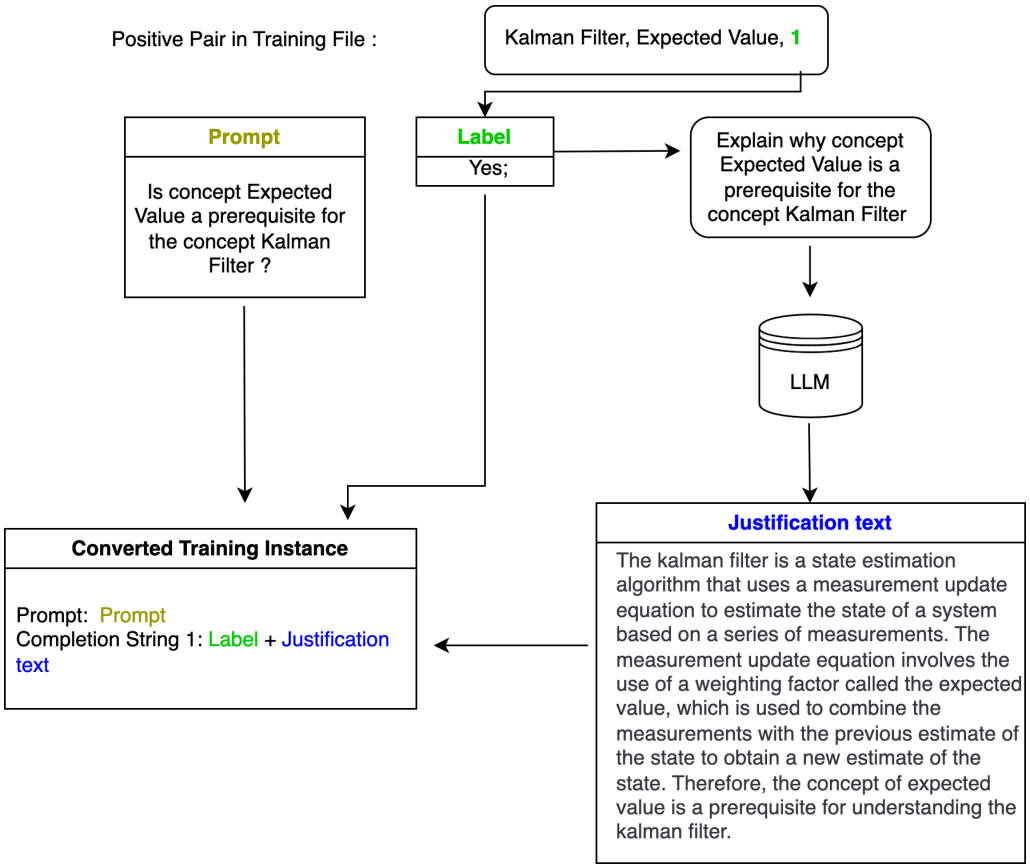

Figure 4.1 Construction of a positive training instance

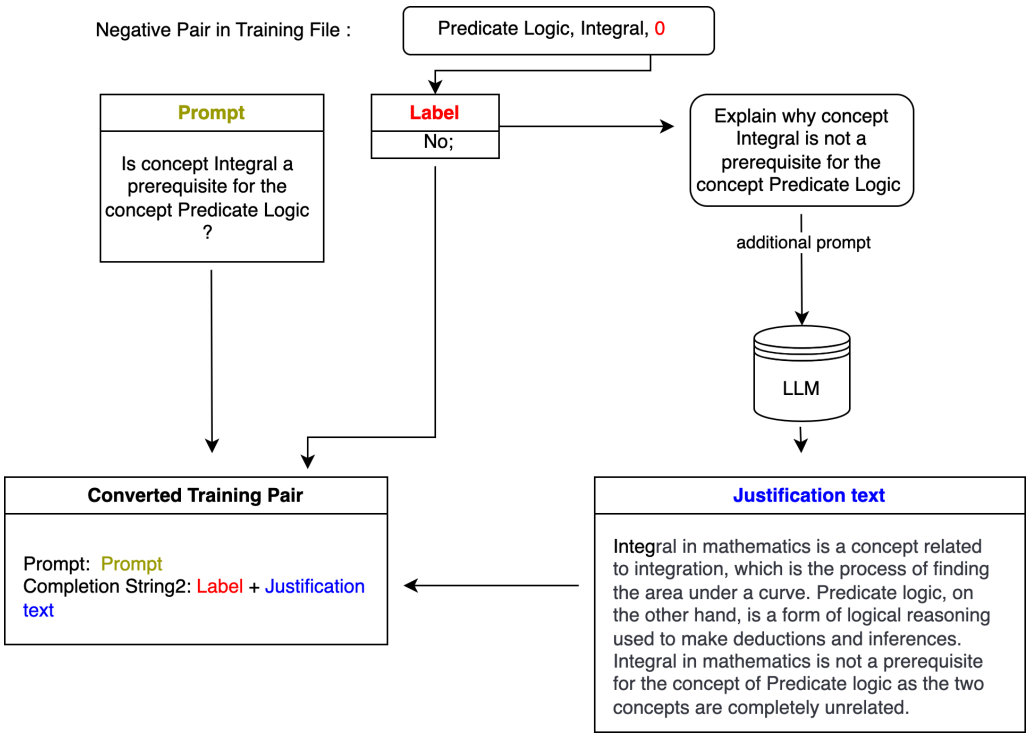

Figure 4.2 Construction of a negative training instance

4.4 Parameter Details for Fine Tuning and Testing

This section introduces two sets of fine-tuning parameters associated with LLMs. The first set of parameters can only be set during the fine-tuning process and second set of parameters is utilized during the testing of the fine-tuned models.

The first set of parameters includes:

- Batch Size: The number of examples used in each iteration of the fine-tuning process. Larger batch sizes decrease the fine-tuning time as the models process data in batches (more than one example at a time). However, increasing the batch size may lead to overfitting as the models fine-tuned with large batches tend to generalize poorly to unseen test data. In GPT-3's official documentation OpenAI (2023), the batch size is recommended to be 0.2% of the number of examples in the training file and we adhered to this guidline during our experiments.

- Learning Rate: This parameter determines the rate at which models update their knowledge during the fine-tuning process. For both of our LLMs, we utilized a learning rate of 5×10^{-2}. According to the official documentation it is stated that larger learning rates often yield better performance in conjunction with larger batch sizes. However, excessively increasing the learning rate can cause the models to make overly large updates to their knowledge, potentially leading them to diverge from the optimal learning path.

- Number of Epochs: The number of times the entire training dataset is scanned during fine-tuning. We set the number of epochs to 3 for both LLMs.

- Maximum Sequence Length: The maximum number of tokens models can process in a single input sequence. We set the maximum sequence length to 128 for both LLMs as the explanations on the prerequisite relations are intended to be short and precise.

The second set of parameters includes:

- Temperature: The temperature parameter controls the level of randomness in LLM outputs. A higher temperature value leads to more diverse and exploratory outputs, while a lower temperature value leads to more conservative and predictable outputs. The temperature parameter directly influences the model's output randomness, reducing uncertainty in predictions by favoring more deterministic responses. Since in the classification task we want definite

answers starting with "Yes" or "No", we set the temperature to 0 during the testing of the fine-tuned models.

- Top P: The top P parameter controls the number of most likely tokens to consider in the model's output. A higher top P value leads to a narrower set of options, while a lower top P value leads to a wider set of options. Similar to *temperature* parameter, we also set $top\ P = 1$ since we aim to ensure the answers follow a specific, expected format.

- Frequency Penalty: The frequency penalty parameter controls the degree to which the model penalizes the repetition of tokens in its output. A higher frequency penalty value leads to less repetition, while a lower frequency penalty value allows for more repetition. We set $frequency\ penalty = 0$ as we have two options for each output: "prerequisite" and "non-prerequisite" and many concept pairs can be, or not be, prerequisites of each other for the same reasons.

4.5 Fine Tuning Strategy of LLMs for Other Tasks in Education

Even though our study mainly focuses on applying the fine-tuning strategy for prerequisite detection task, it is important to recognize that this methodology is not limited to this specific scope. This flexible strategy can be adapted to address a wide range of challenges in various learning settings, thus improving the capabilities of intelligent education systems. To demonstrate this, we first present how this approach can be beneficial in student assignment evaluation. Given the existing records of students' assignments, their respective grades, and instructors' notes on the strengths and weaknesses of each work, LLMs can be fine-tuned to classify new student assignments. They can learn to provide feedback on each assignment's quality and assist students by identifying their mistakes in the assignments. To demonstrate this approach, we present a scenario in Figure 4.3, which outlines the process of tailoring LLMs for assignment evaluation. The fine-tuning procedure starts with the graded student assignments. These assignments include questions, students' answers for the questions, grade of the student and the instructor feedback. The numeric grade information is categorized into n number of classes where n is a pre-determined number set by the instructor. For instance, in the demonstrated schema $n = 5$ means that model will separate students into 5 different classes according to their grades. If the minimum possible grade is 0 and the maximum is 100,

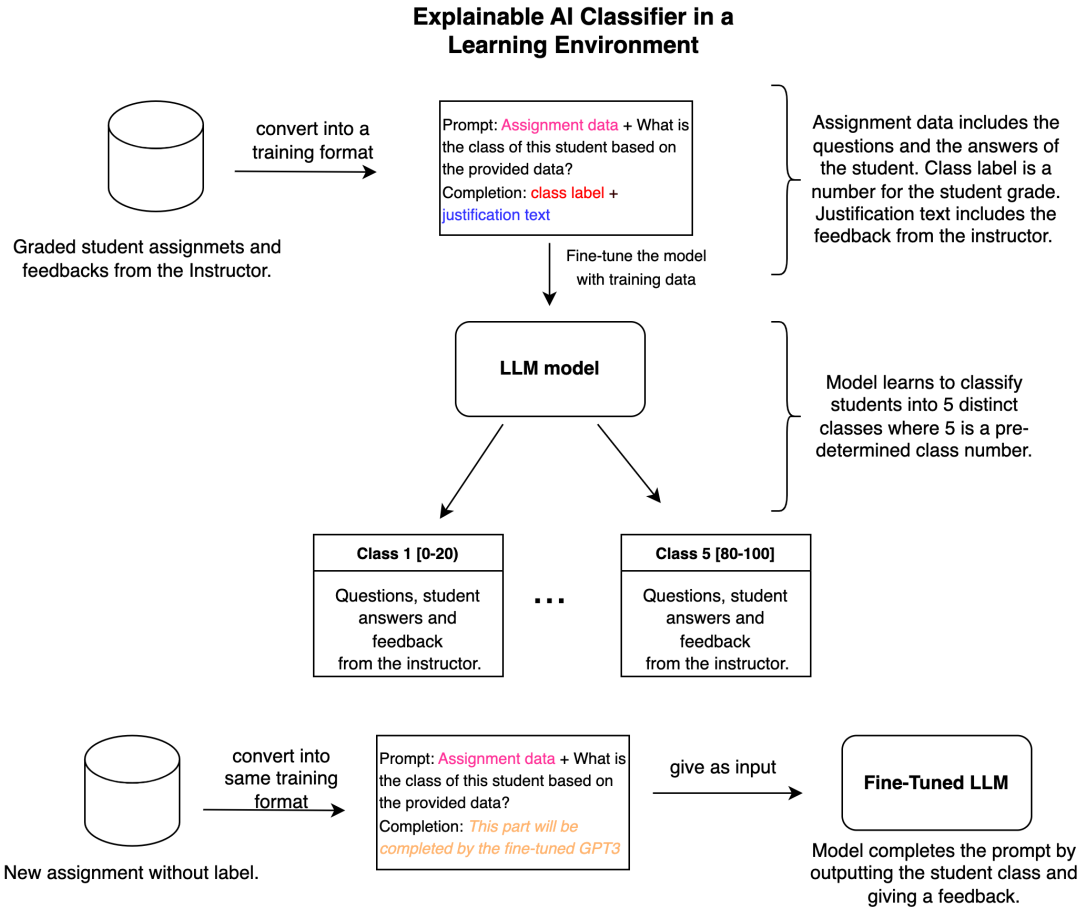

Figure 4.3 The fine-tuned LLM model classifies student assignments, provides feedback for improvement, and explains the rationale for each classification.

the range of grades will be divided into five equal-sized intervals. This will generate the classes such as class 1 [0-20), class 2 [20-40), class 3 [40-60), class 4 [60-80), and class 5 [80-100]. Each class corresponds to a level of performance. Moreover, in order to explain why a certain student is assigned to a certain class, the model will learn to use instructor feedback as a justification for its choice. Most likely, students belonging to class 1 will have a negative feedback and there will be a text explaining the mistakes of the student. Upon a successful fine-tuning, if LLM sees a similar poor assignment paper as the ones in the training, it will be able to classify this paper as belonging to class 1 and generate feedback similar to that of the instructor. Following the illustration of assignment evaluation, we turn to student profiling as another area where the proposed fine-tuning strategy can be effectively employed. In student profiling, the fine-tuned LLM model can be used to understand students' learning behavior. This includes how they engage with classroom materials, their problem-solving speed, their learning pace, preferred learning materials, and more. The fine-tuned model can help categorize students into different groups. More importantly, it can provide reasons as to why a particular student fits into a specific

group. This can allow instructors to understand each student's learning style, tailor content to individual needs and optimize the educational outcomes.

4.6 Integrating LLMs to MEKGs

This section demonstrates how $MEKG$s can be enriched using the outputs of the fine-tuned LLMs. In Figure 4.4, we observe an instance of a $MEKG$ produced by our web application. The nodes are colored and certain edges contain labels. These labels are produced by the fine-tuned GPT-3 model and they explain why a prerequisite relation exists from one node to the other. The explanations for prerequisite relations can either be generated after the $MEKG$ is constructed or during its construction. When an expert is presented with a concept pair (A, B) in ACE algorithm, the prediction of a fine-tuned LLM on why a prerequisite relation exists or not exists from A to B can also be helpful to expert.

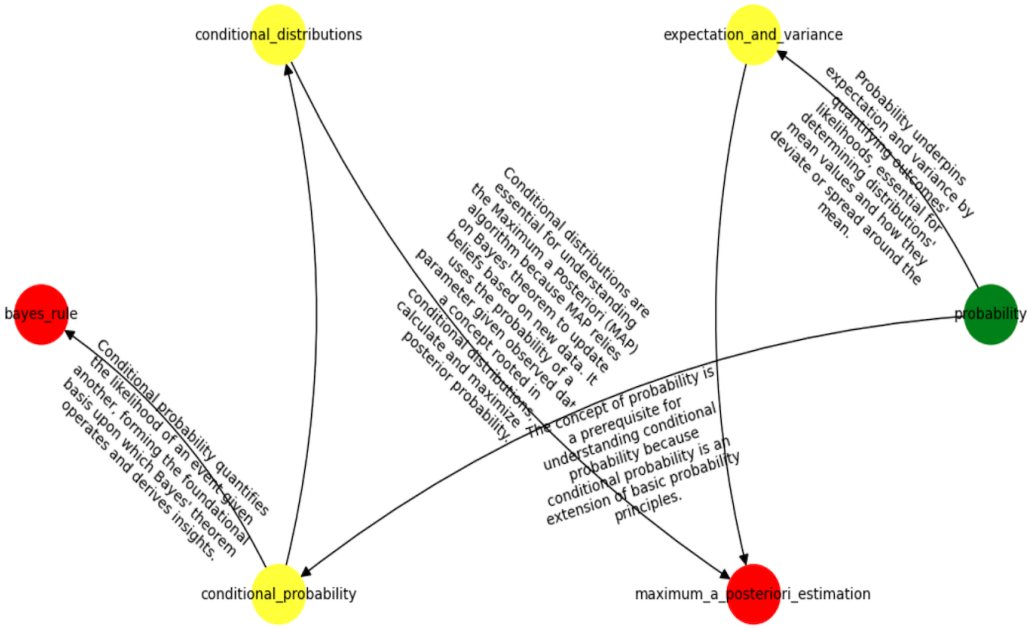

Figure 4.4 Example of a constructed $MEKG$ with node colors and edge labels. A node which has no incoming edge is colored to green indicating that it is a basic concept without any prerequisites and nodes without outgoing edges are colored to red indicating they are advanced concepts.

Generative capability of LLMs can also be used to create automatic descriptions for the concepts in the $MEKG$. Given a general LLM (without any fine-tuning) and a $MEKG$, we design an algorithm called Description Generator (DG) which generates

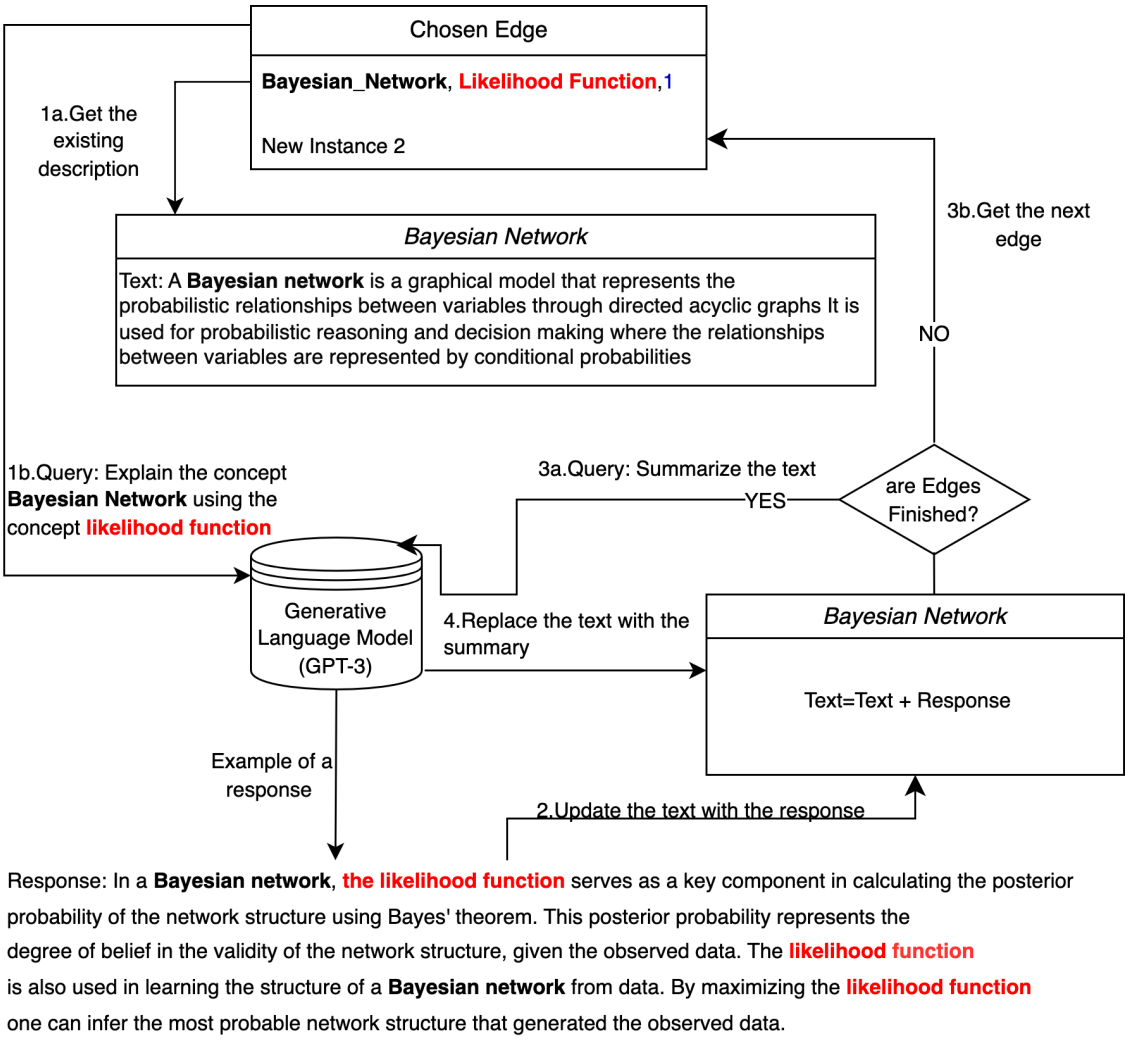

Figure 4.5 An instance of an update in the description of a concept.

an explanation e_A for a concept A where e_A contains all the direct prerequisites of A in the $MEKG$. Therefore, whenever learners read the description of A, they not only understand what it means but also establish a foundational link to the concepts that should ideally be understood prior to studying A. The process of DG is illustrated in Figure 4.5. An edge with concept nodes (Bayesian Network, Likelihood Function) is chosen from the $MEKG$. The description of Bayesian Network is expanded by a text from a LLM which is generated in response to a query "Explain concept Bayesian Network using concept Likelihood Function". If a concept c has n prerequisites, n number of texts are generated in the description. Since the generated descriptions are independent of each other, we unite them in one coherent text with an additional query. This query gets all the generated texts so far and then asks LLM to summarize it. The summarization helps to ensure that the final description is not only comprehensive but also logically consistent and flows naturally from one point to the next.

The change in the descriptions also effect the pairwise CSR scores in a way that if A is a prerequisite of B and C is not a prerequisite of B, then $CSR(B, A) > CSR(C, A)$. This is because after the updates, we make sure that A is semantically referenced more in B's description than in C's description.

4.7 Active Learning Strategy in ACE Algorithm

Recall from Section 3.6 that the ACE Algorithm generates a ranked list (RL) of concept pairs (A, B). Each pair is ranked based on the likelihood that concept A is a prerequisite to concept B. The algorithm presents a concept pair from RL to an expert for evaluation if its relationship cannot be inferred from previously gathered expert responses. Initially, this ranked list is static; it does not dynamically update based on new inputs from the expert. Consequently, once established, the list remains unchanged throughout the expert's interaction with the system, and only those pairs that can be inferred from prior answers are skipped to speed up the annotation process. In this section, we explore how the automated description generation process, outlined in the previous Section 4.6, can be used to dynamically update RL within an active learning framework whenever new feedback is received from the expert.

We show the ACE with Active Learning in Algorithm 3. First, the initial descriptions are used to assign CSR scores to the concept pairs. When the highest scored pair is presented to the expert, if the expert confirms a prerequisite relation, then the description is updated using a LLM. The update carried out by the LLM involves generating a new version of the text description for the concept that now incorporates references from its prerequisite concept. After updating the description, the algorithm readjusts the associated CSR scores. These scores assess the likelihood of prerequisite relations among concepts considering the newly updated descriptions. Such updates lead to a re-ranking of pairs in the list RL, thereby continuously refining which pairs should be presented next to the expert for validation or further inquiry.

Algorithm 3 ACE Algorithm with Active Learning

Input:

$C = \{c_1, \ldots, c_m\}$ — the set of concepts

$d = \{d_1, \ldots, d_m \mid d_i \text{ describes } c_i \in C\}$ — textual descriptions of concepts

$P = \{\}$ — set of direct positive pairs, initially empty

$P_0 = \{\}$ — set of direct negative pairs, initially empty

$LLM =$ selected Large Language Model — the LLM used for updating concept descriptions.

Output:

MEKG for set C

$RL \leftarrow \text{rank}(C, d)$ ▷ Rank all the ordered pairs of C with their initial descriptions.
$EKG = (V, E)$, where $V = \emptyset$ and $E = \emptyset$ ▷ Initial graph with no nodes or edges.
for $k = 0$ **to** $\text{length}(RL) - 1$ **do**
    $(c_i, c_j) = \text{First}(RL)$ ▷ Retrieve the first pair from the ranked list.
    if not isInference(EKG, (c_i, c_j)) **then**
        $answer \leftarrow$ getExpertResponse() ▷ Expert input is required.
        if $answer = 0$ **then**
            $P_0 \leftarrow P_0 \cup \{(c_i, c_j)\}$
            Update(EKG, P_0) ▷ Update the graph knowing no prerequisite exists.
            Reduce(EKG) ▷ Remove transitive edges.
        else if $answer = 1$ **then**
            $P \leftarrow P \cup \{(c_i, c_j)\}$
            Update(EKG, P) ▷ Update the graph knowing a prerequisite exists.
            Reduce(EKG) ▷ Remove transitive edges.
            UpdateDescription(LLM,i, j) ▷ Update description d_j using concept c_i.
            for all $x \in C \setminus \{j\}$ **do**
                UpdateCSR(j, x) ▷ Update CSR score for the pair (j, x).
                $RL \leftarrow \text{rank}(C, d)$ ▷ Re-sort RL
            end for
        else if $answer = 2$ **then**
            break ▷ Expert decided to stop the process.
        end if
    end if
end for

5. Experimental Evaluation

5.1 Introduction

Experimental evaluation consists of two main parts. In the first part, we assess the accuracy of the CSR scores on benchmark prerequisite datasets and report our findings. Following this, we explore the potential advantages of our scoring methodology within educational settings. This examination involves conducting a series of empirical tests using real-life student datasets to observe the correlation between CSR scores and student success. After that, we assess the quality of our constructed $MEKG$s by comparing them to the manually built gold standard graphs. Concluding the first part of our experimental evaluation, we also delve into the practical aspects of the ACE algorithm's performance. Although the theoretical computational complexities of the ACE algorithm are detailed in Section 3.9, we extend our analysis to include practical evaluations. These evaluations consist of specifically designed experiments that monitor how the run-time of the algorithm varies under real-world conditions with different initial parameters.

In the second part of the evaluation, we present a detailed examination of how well LLMs can detect prerequisite relations between concepts. This analysis begins by introducing the datasets used in our study. We then proceed to a comparative performance evaluation of the fine-tuned LLMs on those datasets. Our results demonstrate that the fine-tuned GPT-3 shows significant improvements in performance metrics such as the F-score, surpassing previous models. We also compare the outputs generated by the fine-tuned GPT-3 with those from a non-fine-tuned baseline model to highlight the enhancements achieved through fine-tuning. Further, we evaluate the quality of explanations generated by the fine-tuned GPT-3. We employ a novel method that compares these AI-generated explanations to those crafted by humans, aiming to quantify how well the LLM's outputs align with human explanations. Ad-

ditionally, an in-depth analysis of various parameter configurations used during the fine-tuning process of GPT-3 is conducted. This section emphasizes the critical role careful parameter optimization plays in maximizing model performance. Finally, we include a comprehensive cost analysis covering the computational resources, time requirements, and financial investments necessary for fine-tuning GPT-3 and the other LLM.

5.2 Prerequisite Scoring Methodology as a Supervised Binary Classifier

In this experiment, we create a supervised binary prerequisite classifier $CSR\_bin(t)$ which turns the CSR scores of concept pairs into binary classes 0 and 1 according to a learned threshold t. Class 1 is composed of pairs (A, B) where there is a prerequisite relation from A to B and Class 0 is composed of pairs where there is no prerequisite relation from A to B. By creating a binary supervised classifier, we are able to evaluate our prerequisite scoring methodology against the other well-known approaches in the literature. The parameter t is a learnable parameter and we decide that it should be the t value which gives the highest evaluation metrics in the training data.

5.2.1 Evaluation Metrics for Binary Classification

In the context of a binary classification task, such as identifying if there is a prerequisite relation from one concept to the other, the performance of a classification model is assessed through four common metrics: precision, recall, F1-score, and accuracy. Each of these metrics offers a unique perspective on the effectiveness of the model, providing insights into its strengths and weaknesses. Understanding these metrics is vital for optimizing model performance and ensuring its practical utility in educational applications or other domains where binary classification is employed. Precision measures the accuracy of positive predictions made by the model. It is defined as the ratio of true positive predictions (correctly identified prerequisites) to the total number of positive predictions (both correctly and incorrectly identified prerequisites). High precision indicates that when the model predicts a prerequisite relation, it is likely to be a correct prediction. This is important when the cost of

a false positive (wrongly assuming a prerequisite relation) needs to be minimized. Precision is calculated as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

where TP (True Positives) are instances correctly identified as having a prerequisite relation and FP (False Positives) are instances incorrectly identified as having a prerequisite relation when there is none.

Recall, also known as sensitivity, measures the model's ability to identify all the actual positive cases in the dataset. It is the ratio of true positive predictions to the actual number of positives in the data (the sum of true positives and false negatives). High recall is essential in scenarios where failing to detect a prerequisite relation could have negative implications. Recall is calculated as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where FN (False Negatives) are actual prerequisite relations that the model failed to identify.

The F1-score is the harmonic mean of precision and recall. It provides a single metric that balances both the precision and recall of a model, which is crucial when there is a need to find a compromise between making accurate positive predictions and ensuring no positives are missed. The F1-score is particularly useful in situations where an imbalance between the classes might render other metrics less informative. F1 score is calculated as follows:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Accuracy measures the overall correctness of the model across both positive and negative predictions. It is the ratio of correct predictions (both true positives and true negatives) to all predictions made by the model. It is a simple metric which can be preferable in cases where there are equal numbers of positive and negative instances in the dataset. Accuracy score is calculated as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

We decide to use t value that gives the best F1 score in the training data, as both

recall and precision are equally important in this task.

5.2.2 Utilized Dataset and Compared Models

We evaluate $CSR\_bin(t)$ on University Course Dataset (UCD), introduced by Liang, Ye, Wu, Pursel & Giles (2017). UCD dataset initially contained 1008 manually annotated prerequisite concept pairs which are extracted from the Computer Science course syllabus of various universities in the USA. UCD was later enriched by Roy et al. (2019) by providing 1512 negative instances (i.e., non-prerequisite pairs) on top of 1008 positive pairs making it a larger and more complete dataset with 2520 pairs. The dataset can be downloaded from Github [1]. Roy et al. (2019) conducted a comparative analysis of the outcomes associated with four distinct prerequisite detection strategies applied to this dataset. Two of these strategies, developed by Pan et al. (2017) and Liang et al. (2017), are referred to as **MOOC-RF** and **CPR-Recover** and the remaining two strategies, introduced by Roy et al. are referred to as **PREREQ** and **Pairwise LDA**. Each concept pair (A, B) in UCD is labeled either 0 or 1. Label 1 indicates that A is a prerequisite of B, and 0 indicates that A is not a prerequisite of B. Each concept in the concept pair has its corresponding Wikipedia article; therefore, the textual content in these articles is used as concept descriptions. In addition to these four mentioned models, we also create a baseline GPT-3 model through zero-shot prompting to test if raw large language models are capable of understanding prerequisite relations. GPT-3 takes an input string from the user which is called a *prompt* and produces an output string called a *completion*, which contains a sequence of words that are most likely to come after the *prompt* according to the model's trained probabilistic associations. In order to test GPT-3 as a baseline binary classifier, we turn each instance in UCD test dataset to a prompt as depicted in Figure 5.1. Given a prompt for a concept pair (A, B), if the *completion* string contains the keyword "Yes", we label the instance as 1; otherwise, we label it as 0.

Liang et al. report that they use the 60% of UCD data as training and 40% of it as testing for all the reported models. In order to make a fair comparison, we also use the same ratio. After training, we learn the t value to be 68. As for the GPT-3 model, we don't do any additional training or fine-tuning and use it as a baseline.

[1] https://github.com/suderoy/PREREQ-IAAI-19

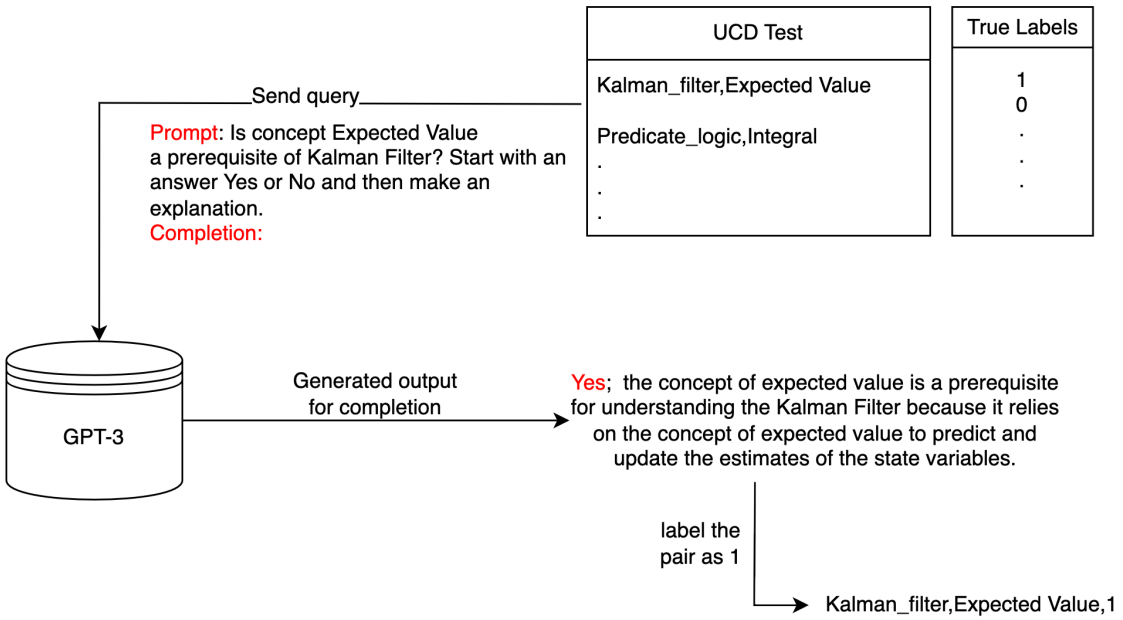

Figure 5.1 Zero shot GPT-3 used as a binary classifier

Table 5.1 Table comparing different prerequisite detection methods.

Name of the method	**Precision**	**Recall**	**F1 score**
PREREQ	46.76	91.64	59.68
Pairwise LDA	98.27	16.42	28.14
CPR-Recover	16.66	46.51	24.54
MOOC-RF	43.70	53.43	50.95
CSR binary classifier(t=68)	46.13	**1**	**66.53**
Zero-shot GPT-3	84.10	35.33	49.65

In Table 5.1, we compare the performance of $CSR\_bin(t=68)$ with five other models: PREREQ, Pairwise LDA, CPR-Recover, MOOC-RF, and Zero-shot GPT-3 in terms of recall, precision, and F1 score. From the table, we can observe that $CSR\_bin(t=68)$ outperforms other models in terms of the F1 score. It achieves an F1 score of 66.53, which is higher than the second-best method, PREREQ, with a score of 59.68. In terms of recall, our method achieves a perfect score of 1, meaning all actual prerequisite pairs in the dataset were correctly identified by labeling top %68 of the CSR-sorted pairs as prerequisites. When it comes to precision, $CSR\_bin(t=68)$ outperforms PREREQ and MOOC-RF by achieving a precision of 46.13 which means that it is more accurate in identifying true positives. However, it is outperformed by Pairwise LDA and Zero shot GPT-3 which show a high

precision of 98.27 and 84.10 respectively.

Overall, we conclude that simultaneously achieving high precision and recall in the context of binary prerequisite detection tasks still remains a significant challenge. Nonetheless, our methodology demonstrates a promising approach, particularly in maximizing recall without substantially compromising precision, thereby contributing to more effective and reliable identification of prerequisite relationships in educational content.

5.3 Testing the Predictive Performance of Prerequisite Scoring

Methodology on Student Success

In this experiment, we test if CSR scores are an indicator of student performance. Remember that $CSR(B, A)$ is high if concept A is highly referenced in the description of concept B. In case of a high $CSR(B, A)$ score, we expect that students who already know concept A perform better on concept B when compared to the control group. Conversely, in case of a low $CSR(B, A)$ score, students knowing A are not expected to have better performance on B when compared to the control group. Control group is a set of randomly selected students whose knowledge on A is not known.

In order to test our hypothesis, we use the student logs from a dataset of a real-world educational platform obtained from the work of Gong, Smith, Wang, Barton, Woodhead, Pawlowski, Jennings & Zhang (2022). We partition the dataset into two tables *concept_metadata.csv* and *student_answers.csv*. The *concept_metadata.csv* contains 756 different secondary school mathematics concepts with columns:

- **Concept IDs:** A unique numerical code assigned to each concept.
- **Concept Name:** A descriptive title associated with each concept.

The *student_answers.csv* table contains 6468 unique students and their answers for questions related to these concepts with columns:

- **User IDs:** A unique identifier for each student.
- **Concept and Question IDs:** Identifiers that link each question to its corresponding concept.

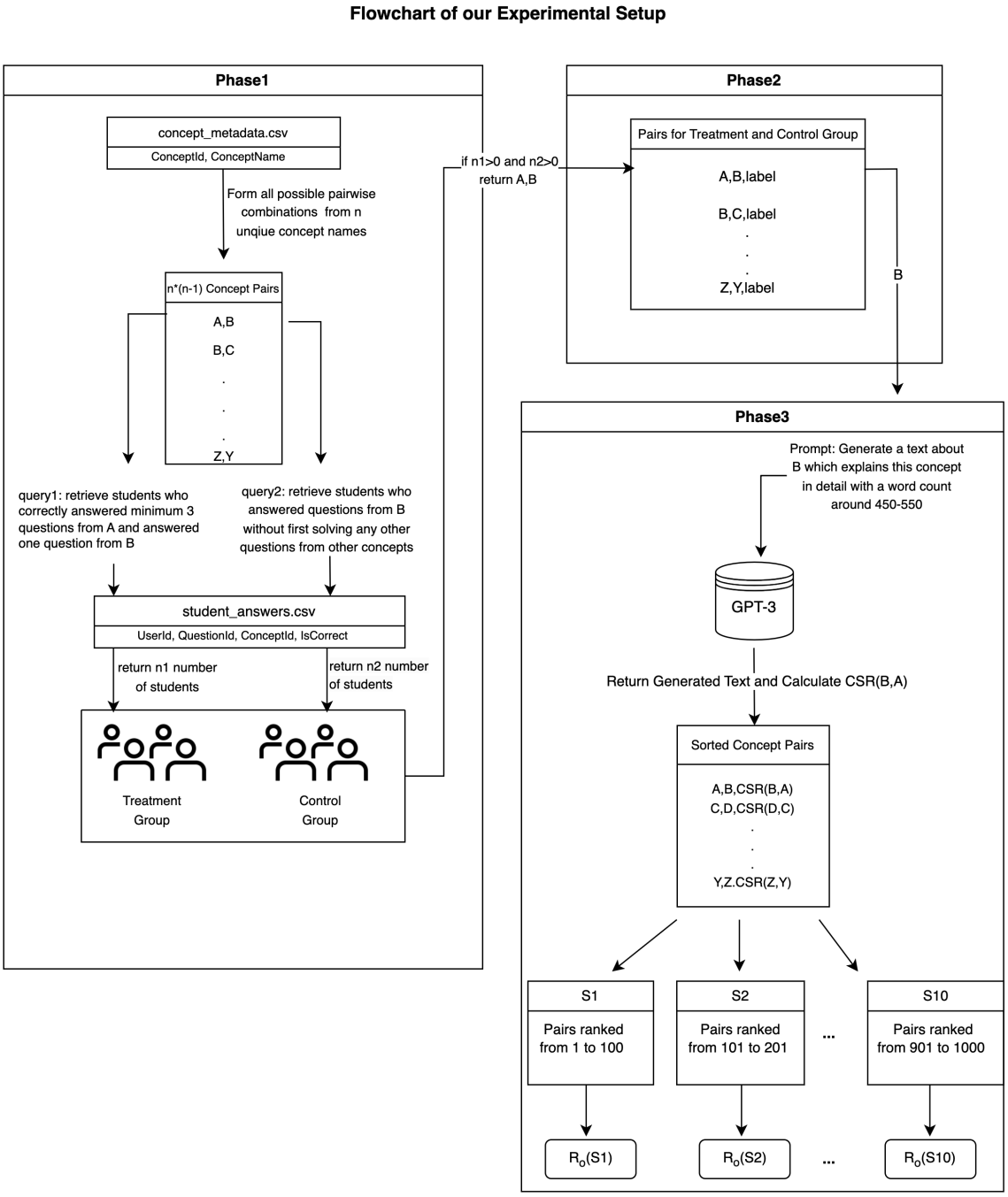

Figure 5.2 Overview of Our Three-Phase Experimental Setup

- **Timestamps:** The exact date and time of each attempt on a question.
- **IsCorrect:** A binary value indicating if the student's answer to a question is correct.

Questions are multiple-choice with 4 options A, B, C, and D. We assume that a student knows a concept, if she correctly answered at least 3 different questions related to that concept at her first attempt. Our experiment has three phases as depicted in Figure 5.2. In Phase 1, we find all possible pairs (A, B) from the initial 756 unique concepts such that we can form a Treatment Group TG consisting of students who know A and solved questions related to B, and a Control Group CG consisting of students who only solved questions related to B. In Phase 2, we calculate the students' average ratio of correct answers for the questions related to B for both groups to determine if TG outperforms CG. In Phase 3, we sort concept pairs formed in Phase 1 based on their CSR scores to see how the average ratio of correct answers in TG and CG change for decreasing CSR scores.

In Phase 1 of our experiment we form all ordered concept pairs (A, B) from the initial 756 concepts which corresponds to $(756 \times 755 = 570780)$ concept pairs. For each pair in the list, we check if both TG and CG are formed. TG is formed for pair (A, B) if we can find students who correctly answered minimum 3 questions from A in their first attempt and answered at least one question from B. CG is formed if we can find students who solve at least one question from B without first solving questions from any other concept. If both TG and CG are not formed for a pair of concepts, we discard that pair and move on to the next pair in the list. Following this methodology, we formed a test set of 1000 concept pairs for which the average number of students in TG and CG are 15.75 and 27.28 respectively.

In Phase 2, for each of the 1000 concept pairs (A, B), we calculate students' average ratio of correct answers to B in TG and CG. If the average ratio of TG is higher than CG, we label that pair as 1, indicating that TG outperforms CG and label it as 0 if CG outperforms TG. Given a sequence S of concept pairs, we count the number of concept pairs in S for which TG outperforms CG (i.e., pairs with label 1) and define the ratio of outperforming pairs to all pairs $R_o(S)$ formally as:

$$R_o(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} \text{label}(S[i])$$

where $label(S[i])$ is the label of the ith concept pair in S, and $|S|$ is the number of concept pairs in S.

The third and final phase of our experiment begins with the calculation of CSR scores of the 1000 concept pairs (A, B) that we obtained in Phase 1. Our benchmark data set does not contain the textual descriptions of concepts, therefore in order to calculate CSR scores through semantic references, we opted to use GPT-3 to generate the textual descriptions. We then calculate CSR scores based on those descriptions and sort the concept pairs (A, B) in descending order of $CSR(B, A)$. The sorted list of 1000 concept pairs is then partitioned into 10 subsequences $S_1, S_2, ..., S_{10}$, each containing 100 concept pairs such that S_1 has the top 100 concept pairs, S_2 contains the next 100 concepts and so on. Finally, we calculate $R_o(S_j)$ for each of S_j for $1 < j < 10$.

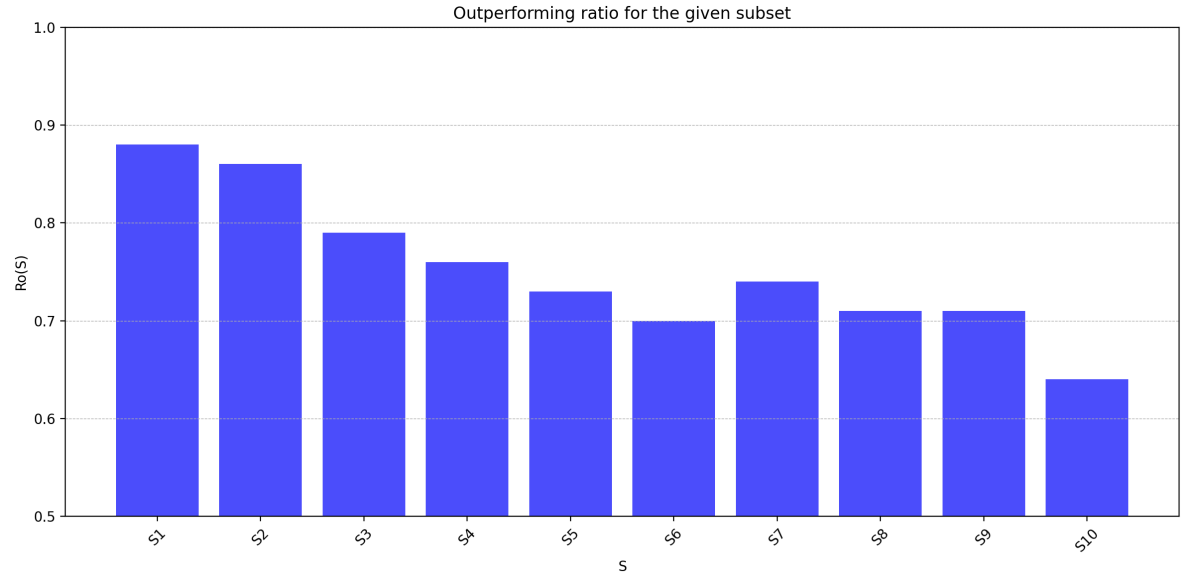

Figure 5.3 Outperforming ratio of TG over CG for lists of concept pairs with decreasing CSR scores

In Figure 5.3 we plot the ratio of outperforming pairs of concepts where S_1 denotes the list of concept pairs with the highest CSR scores and S_{10} contains the pairs with the lowest CSR scores. As can be seen in Figure 5.3, we have the highest ratio of outperforming concept pairs for S_1 with $R_o(S_1) = 0.88$. The ratio falls as the CSR scores decrease. We see the lowest ratio $R_o(S_{10}) = 0.64$ for S_{10} containing pairs with the lowest CSR scores.

If knowing the semantically referenced concepts has zero effect on the performance, the outperforming ratio is expected to be close to 0.5. However, our benchmark data set consists of concepts only within the Mathematics domain, therefore all the concepts are related and even knowing the least semantically referenced concept has some positive effect on the student performance, this could be the reason for all the R_o values being above 0.5.

Overall, the decreasing R_o values in the lower ranked pairs (A, B) is an indication that students knowing the most semantically referenced concepts in the descriptions of B tend to perform better on the questions related to B compared to a random student's performance on the questions related to B, meaning that our CSR scoring methodology can be a predictor for student success. Therefore, EKGs constructed using CSR scores may guide students on their learning journey.

For reproducibility, we have posted all the data sets we extracted from the original benchmark data set on Github [2]. The Github page includes all the Treatment Groups (TG), Control Groups CG, concept pairs, textual descriptions (obtained from GPT-3) as well as the CSR scores.

5.4 Constructing and Evaluating MEKGs: Methodologies, Quality Metrics, and Efficiency Factors

In this section, we construct $MEKG$s using our ACE methodology defined in Algorithm 1 and assess the quality of the resulting graphs. To evaluate the produced $MEKG$s, we use three different gold-standard datasets in our experiments. The first dataset is an KG from an e-learning platform called Metacademy which is specialized on topics related to machine learning[3]. This graph, which we refer to as Metacademy, has 141 nodes, each representing a concept from Machine Learning, Statistics, or Linear Algebra. Each concept has a short description provided by the experts of the Metacademy platform where the directed paths represent the prerequisite relations among concepts. We remove the transitive edges from Metacademy graph turning it into a $MEKG$. We also create our own gold-standard $MEKG$ using our ACE web-application. For that, we choose the Data Structures and Algorithms field, one of the core disciplines of computer science, and named the corresponding graph, consisting of 29 concepts, as DSA.

We use the gold-standard $MEKG$s with *path recall* and as *path precision* as the quality metrics to compare two graphs. Let G_1 and G_2 represent two $MEKG$s that we want to compare. Let P_1 be the set of all paths in G_1 and P_2 be the set of all paths in G_2. We define path recall as:

[2] `https://github.com/cemaytekin/EKG-Dataset`

[3] `https://metacademy.org/browse`

$$PathRecall(G_1, G_2) = \frac{|P_1 \cap P_2|}{|P_1|}$$

Path precision is defined similarly:

$$PathPrecision(G_1, G_2) = \frac{|P_1 \cap P_2|}{|P_2|}$$

Assuming that the $MEKG$ produced through the ACE methodology is denoted by $MEKG_A$ and the gold-standard graph is denoted by $MEKG_G$, to demonstrate the role of the expert on the quality of $MEKG_A$, we plot the relative reduction of expert effort on the x-axis and plot the PathRecall($MEKG_G$, $MEKG_A$) on the y-axis. This way we can observe the effect of reduced expert effort (i.e., reducing parameter t) on the path recall. We utilize 20 different values of t ranging from 5 to 100. We assume that the expert correctly identifies the prerequisite pairs in the top t% of the concept pairs. Therefore the *path precision* which measures the fraction of paths in the compared $MEKG$ that are also in the standard $MEKG$ is always equal to 1 and we do not report it in the experiments. Furthermore, in order to observe the contribution of semantic reference over exact reference in the ranking process, we compute the prerequisite scores for the pairs twice, employing both CSR and CER based approaches. Additionally, we use random ordering of the pairs as a baseline in which the t percentage of pairs is randomly presented to the expert, and the expert constructs the graph from those pairs.

Dataset	# Concepts	# Unordered Pairs	# Direct Prerequisites	Total
Metacademy	141	9870	331	1586
DSA	29	406	55	111

Table 5.2 Description of the two datasets that are used as gold-standard $MEKG$s

In Figure 5.4, we observe the path recall values on the y-axis for different values of relative expert effort reduction. For instance, when the relative expert effort reduction is 50%, the path recall for CSR is approximately 90%, indicating that our methodology produces a $MEKG$ that is 90% similar to the Metacademy $MEKG$ by letting the expert evaluate only half of the pairs in worst case (without the inferences). As expected, the lowest average recall value (0.385) belongs to random ordering. We also see that choosing semantic references over exact references increase average path recall from 0.485 to 0.684 which is a significant improvement. We also show the maximum relative reduction in the plot as a dashed line. This line

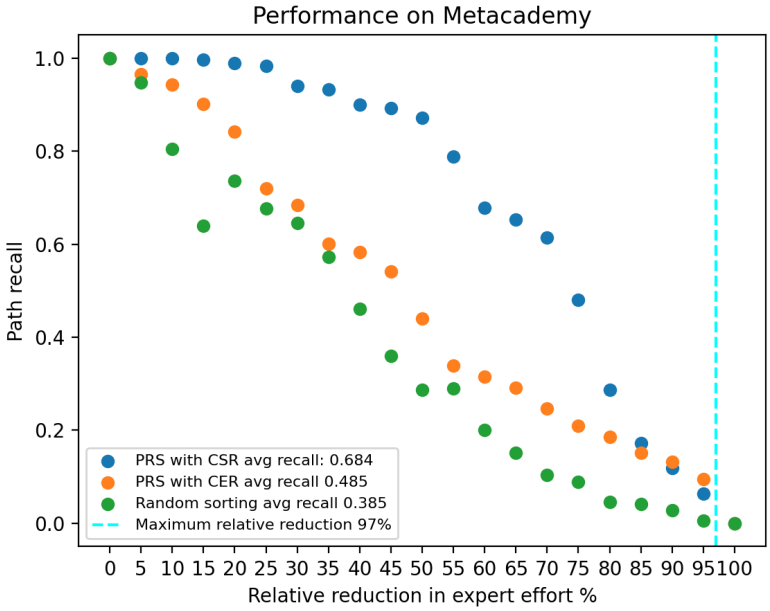

Figure 5.4 Impact of CSR on path recall in Metacademy dataset.

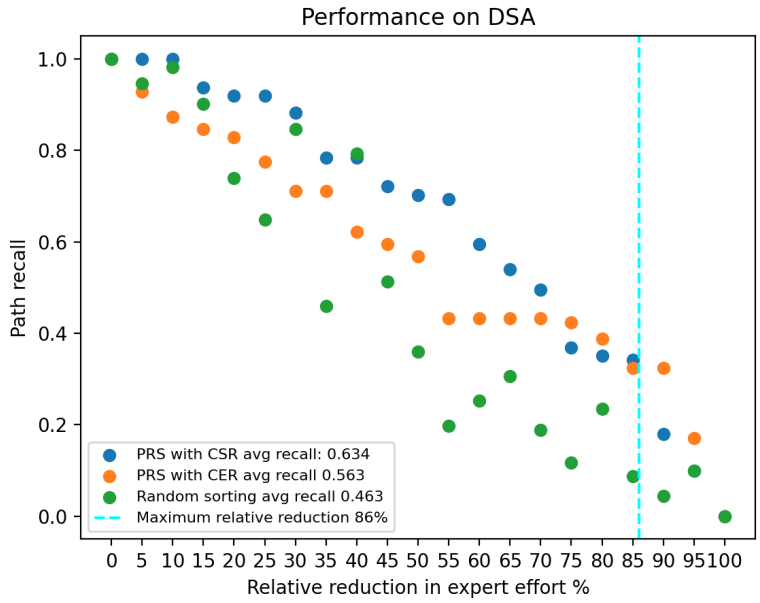

Figure 5.5 Impact of CSR on path recall in DSA dataset.

corresponds to x-axis value of 97 indicating that if the prerequisite ranking algorithm would be perfect, (every presented pair to the expert includes a prerequisite relation) then it would give a path recall value of 1 from 0 to 97 relative expert effort reduction. Therefore, while we can conclude that semantic references help us succeed in making our approach more feasible, there is still room for improvement. Similarly in Figure 5.5, we observe that CSR mode achieves the best performance compared to the other approaches with a score of 0.634. Between the x-axis values

(75-95) we see that CSR and CER show almost equal performances. This can be due to the fact that pairs with strong prerequisite relations tend to possess both exact and semantic references, while pairs with more subtle prerequisite relations typically display only semantic references. As the relative reduction in expert effort becomes more significant, only the topmost pairs are presented for evaluation. Consequently, both modes CSR and CER adequately capture these significant pairs. However, as the relative reduction in expert effort becomes less substantial, CSR outperforms CER by effectively differentiating between subtle prerequisite pairs and non-prerequisite pairs.

We also test the effect of the language model selection on the resulting quality of the constructed $MEKG$. In order to do that, we construct 3 different $MEKG$s one constructed with our main language model all-MiniLM-L6-v2 and other two constructed using Word2Vec and Fasttext. For each constructed $MEKG$, we calculate path recall for different relative reductions in expert effort and show the results in Figure 5.6. It can be observed from the figure that with all-MiniLM-L6-v2, we have higher path recall in the constructed $MEKG$ for every t value between 5 to 95 indicating that it brings the top t percent of the sorted pairs to expert more accurately than the other language models.

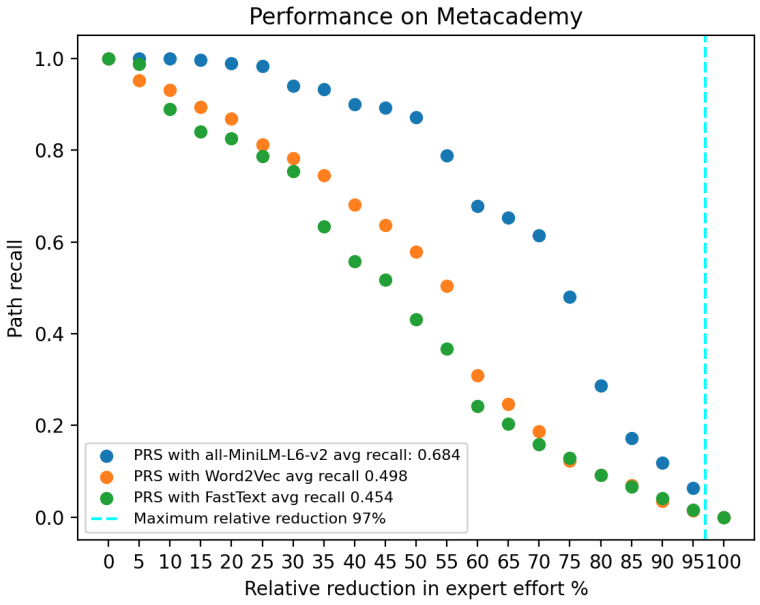

Figure 5.6 Impact of utilizing different language models for CSR.

There are three main factors effecting the runtime of our methodology: (1) Length of the concept descriptions, (2) The size of the concept set, and (3) The choice of the language model in CSR mode. To understand the impact of the first factor, we prepared five ranked lists for DSA with varying lengths and recorded the time

taken to prepare each list, with the results detailed in Table 5.4. From the table, we observe that as the length of the concept descriptions increases, the runtime of our methodology also increases. This is expected as longer descriptions require more processing time to generate all pairwise prerequisite scores. For instance, when the length of the concept descriptions is 108 words, the runtime is 18 minutes, whereas for descriptions with a length of 1370, the runtime increases to 160 minutes.

# concepts in subset MEKG	**Runtime (in minutes)**
30	15
60	55
90	125
120	185

Table 5.3 Effect of the number of concepts on Runtime.

Avg length concept descriptions (# of words)	**Runtime (in minutes)**
108	18
229	35
438	55
838	90
1370	160

Table 5.4 Effect of Concept Description Length on Runtime.

In order to assess the impact of the second factor, we use four different subsets of concepts from Metacademy dataset and construct four different $MEKG$s with different number of concepts (Metacademy n=30, Metacademy n=60, Metacademy n=90 and Metacademy n=120). Each description has on average 78 words. From Table 5.3, it can be observed that as the number of concepts increases, the runtime of the methodology also increases in proportion.

Lastly, in order to understand the effect of third factor, we present Table 5.6 which demonstrates the effects of the usage of different language models on the runtime. The three models analyzed are all-MiniLM-L6-v2, Word2Vec, and Fasttext. The table presents the runtimes in minutes for ranking all the pairwise prerequisites of 141 Metacademy concepts each having an average description size of 78 words.

Based on the results presented in the table, it can be observed that the all-MiniLM-L6-v2 model has the longest runtime of 150 minutes. On the other hand, both Word2Vec and Fasttext have considerably shorter runtimes, with 12 minutes and 9

minutes, respectively. However, they exhibit lower path recall values as shown in Figure 5.6.

Utilized Word Embedding Model	Runtime (in minutes)
all-MiniLM-L6-v2	150
Word2Vec	12
Fasttext	9

Table 5.5 Effect of word embedding model on runtime on the Metacademy dataset.

These findings suggest that while all-MiniLM offers improved quality in the constructed $MEKG$s compared to Word2Vec and Fasttext, it constitutes the main bottleneck in the runtime of the methodology.

Utilized Word Embedding Model	Runtime (in minutes)
all-MiniLM-L6-v2	150
Word2Vec	12
Fasttext	9

Table 5.6 Effect of word embedding model on runtime on the Metacademy dataset.

5.5 Evaluating LLMs

The second part of the evaluation analyzes the prerequisite detection performance of the fine-tuned LLMs. We do this analysis on three different datasets. The first dataset is University Course Dataset (UCD) which is introduced in Section 5.2.2. The second dataset is Course Dataset (CD) and the third is MOOCCubeX Dataset (MC). Two fine-tuned LLMs' performance (GPT-3 and LLAMA2) are compared on UCD. We show that the fine-tuned GPT-3 is able to outperform other binary prerequisite classifiers with a 9% improvement in F-score. Next, outputs generated by the fine-tuned GPT-3 are compared against the raw, non-fine-tuned GPT-3 and observed differences are reported. After that, the performances of the fine-tuned GPT-3 and LLAMA2 are tested on CD and MC datasets. In addition to these assessments, we put forward a unique method of comparison between AI-crafted explanations and human explanations to measure the LLM's explanatory performance. Furthermore, we also conduct an analysis of various parameter configurations for fine-tuning LLMs. The analysis underscores the importance of careful optimization of these parameters for achieving better performance. Finally, we provide a

cost analysis for fine-tuning GPT-3 and the LLAMA2 model. Covering aspects like computational resources, time requirements, as well as financial investment, our comprehensive analysis offers a better understanding of the LLMs' practical implications.

5.5.1 Benchmark Datasets

The first benchmark dataset we utilize is the Course Dataset (CD), introduced by Liang et al. (2015) in 2015. The dataset is based on information obtained from a university's course website, and it includes prerequisite relations between courses in the domains of Computer Science and Mathematics. The authors declare that domain experts have checked and corrected the labels of all course pairs. They compare their **RefD** model on this dataset with **MaxEnt**. This dataset was later utilized again for training and testing in the work of Manrique et al. (2019) and the authors reported state-of-the-art results with the **XGBOOST** model. The second benchmark dataset is the University Course Dataset (UCD) which is discussed in Section 5.2.2. The third dataset we employ is constructed by Yu et al. (2021). Similar to the previous two benchmark datasets, MC-PSY contains concept pairs labeled with respect to their prerequisite relation where the domain of the concepts are from Computer Science, Mathematics and Psychology. From MC-PSY we only included concepts from Psychology since concepts from Computer Science and Mathematics are already covered by the other datasets we used in our evaluation. The details of the datasets we used are provided in Table 5.7.

Dataset	**Domain**	**# Pairs**	**# Prerequisites**
Course Dataset (CD)	CS	678	108
Course Dataset (CD)	MATH	658	75
University Course Dataset (UCD)	CS	2520	1008
MOOCCube (MC)	PSY	1000	485

Table 5.7 Benchmark Datasets

5.5.2 Performance Comparison of LLMs to Other Models on UCD

GPT-3 and LLAMA2 are fine-tuned on UCD training data and the results are compared against the previous works in the literature. The results of the four other models **CPR-Recover**, **MOOC-RF**, **PREREQ**, **Pairwise LDA** are taken from the work of Roy et al. (2019), and the result of **MHAVGAE** is taken from the work of Zhang et al. (2022). For a fair comparison, we shuffled all dataset instances, allocating 80% for training and the remaining 20% for testing, following the same procedure used in previous studies. In the evaluation of Zhang et al. (2022), authors do not mention precision and recall scores separately and report only the F1-score on UCD therefore for **MHAVGAE** we are only able to compare the F1-scores. Our method does not require external resources to calculate prerequisite relations; therefore, we achieve consistent results for both UCD and the MOOC dataset, given that the concept pairs in both datasets are identical. However, for methods that rely on external resources, the results differ. This variation occurs because the UCD dataset employs university course syllabi and the MOOC dataset employs video lectures as an external resource to calculate prerequisite relations.

In Table 5.8, we see the performances of the fine-tuned GPT-3 and LLAMA2 on UCD. Fine-tuned GPT3 outperforms the second best model (**MHAVGAE**) by a margin of 9% in terms of F-score. The fine-tuned LLAMA2, while not surpassing the fine-tuned GPT-3 and **MHAVGAE**, still performs better than average, ranking as the third-best model with an F-score of 65.3. In both LLMs, we observe that fine-tuning improves the models' capabilities in detecting prerequisite relations. Specifically, we see that the F-score increases by 37.69% for GPT-3 (from 49.65 to 87.35) and by 19% for LLAMA2 (from 46.2 to 65.3). The best recall value on UCD is achieved with **PREREQ** and the best precision value is achieved with Pairwise LDA. Roy et al. (2019) use course syllabi as an external knowledge base, which include course descriptions and the prerequisite information between courses. Authors apply LDA to construct concept vectors where indices of each vector represent the concept's closeness to each course description. Concept vectors are then given to a deep learning model called **PREREQ** and the model learns the prerequisite relations between concepts using the provided training file. In the case with Pairwise LDA, the method not only constructs concept vectors indicating their closeness to course descriptions but also directly incorporates the provided course-level prerequisite information. By analyzing concept vectors pairwise and integrating this explicit prerequisite data, Pairwise LDA calculates scores that reflect the likelihood of one concept being prerequisite to another. We assume that the essential courses that are first taught in the universities (without any prerequisites) include simple

concepts and courses that have many prerequisites include complex concepts. Given a concept pair (A, B), if Pairwise LDA is able to detect the frequent occurrence of A in the basic courses and the frequent occurrence of B in the course with multiple prerequisites, we may assume that it correctly identifies the prerequisite relation. However given its low recall, we may argue that these types of conditions do not occur frequently. As discussed in Chapter 2, the interpretability of the deep learning models is difficult and authors do not explain why **PREREQ** shows a high recall but low precision. In fact this is one of our motivations for teaching LLMs to provide an explanatory output for their decisions. Therefore, in addition to classification performance, we also analyze the explanations of the raw and fine-tuned GPT-3 in order to understand the models' reasoning on prerequisite relation. In the next section, we present various output samples from raw GPT-3 and fine-tuned GPT-3. Using the generated outputs for each instance, we investigate what the GPT3's initial perception of prerequisite relation was and how it is modified according to the UCD training data after the fine-tuning.

5.5.3 Observable Changes in Fine Tuning on UCD

We manually analyze the outputs of GPT-3 and fine-tuned GPT-3 on random test instances of UCD and present our findings in this section. We can already see from Table 5.8 that raw GPT-3 has a low recall score but high precision score which indicates that the model tends to label some prerequisite pairs as non-prerequisites; however, the ones it labels as prerequisites are correct 84% of the time. Therefore we can assume that model has a strict prerequisite definition before the fine-tuning. For instance, the *prompt* : "*Is Linear Algebra a prerequisite for Machine Learning*" has a *completion*: "*No;Linear Algebra is not a prerequisite for Machine Learning. However, a basic understanding of Linear Algebra is helpful in understanding how Machine Learning algorithms work and how to interpret their results*". The pair (Linear Algebra, Machine Learning) has a label 1. This suggests that the experts who created the dataset consider any concept that can be helpful in understanding the other concept as a prerequisite. When the same test instance is given to fine-tuned GPT-3, we see the *completion*: "*Yes; Linear algebra provides essential tools for working with vectors, matrices, and scalars, which are central to many Machine Learning algorithms*". By comparing the generated outputs, we observe that the model has learned to be more flexible in its definition of prerequisite relations. In order to understand to what extent GPT-3 and fine-tuned GPT-3 know what a prerequisite relation means, we create test cases to determine if both models consistently

Dataset	Method	Precision	Recall	F-score
UCD	PREREQ	46.76	91.64	59.68
UCD	Pairwise LDA	98.27	16.42	28.14
UCD	CPR-Recover	16.66	46.51	24.54
UCD	MOOC-RF	43.70	53.43	50.95
UCD	GPT-3(No fine tuning)	84.10	35.33	49.65
UCD	**Fine-tuned GPT3**	**84.47**	**90.5**	**87.34**
UCD	MHAVGAE	-	-	78.75
UCD	Raw LLAMA2	77	33	46.2
UCD	Fine-tuned LLAMA2	70	61.2	65.3
MOOC Dataset	PREREQ	55.60	75.74	60.73
MOOC Dataset	Pairwise LDA	48.43	10.47	17.22
MOOC Dataset	CPR-Recover	17.18	52.97	25.94
MOOC Dataset	MOOC-RF	59.74	56.48	58.07
MOOC Dataset	GPT-3(No fine tuning)	84.10	35.33	49.65
MOOC Dataset	**Fine-tuned GPT3**	**84.47**	**90.5**	**87.34**
MOOC Dataset	Raw LLAMA2	77	33	46.2
MOOC Dataset	Fine-tuned LLAMA2	70	61.2	65.3

Table 5.8 Precision, Recall, and F-score for Various Methods on UCD Datasets

validate the prerequisite relation's natural properties, as defined in Section 1.2. In order to achieve this, we get all the positive pairs (A, B), reverse their order as (B, A) and ask the model again to classify the instance. According to the asymmetry property, each model should always label such instances as 0. Similarly, we also create custom pairs (A, A) and (B, B) to test the irreflexivity property and finally, if there exists a positive pair (A, B) and (B, C), we ask the label of the instance (A, C) to test the transitivity property. From Table 5.9, we see that raw GPT-3 already knows that a concept can not be a prerequisite of itself and this knowledge is preserved in fine-tuning as well. The fine-tuned GPT-3 demonstrates improved adherence to the Asymmetry and Transitivity rules, with accuracy increasing from 72.2 to 85.7 and from 54.5 to 80.9, respectively. Although our scores significantly surpass those of other models, the fine-tuning data may not have been sufficient for the model to learn the Asymmetry and Transitivity rules perfectly.

Axiom	GPT-3		GPT-3 fine-tuned	
	Number of Instances	Accuracy	Number of Instances	Accuracy
Irreflexivity	400	100%	400	100%
Asymmetry	72	72.2%	147	85.7%
Transitivity	11	54.5%	21	80.9%

Table 5.9 Prerequisite Axioms

Method	CS-Accuracy	CS-Precision	CS-Recall	CS-F1
MaxEnt	72.8	87.6	53.2	66.1
RefD-EQUAL	76.4	80.4	69.9	74.7
RefD-TFIDF	77.1	82.3	69.1	75.1
XGBOOST	81.3	90.8	83.1	86.8
GPT-3	62.2	93.7	26.3	41
Fine-tuned GPT3-(1)	72.2	86.9	78.9	80.7
Fine-tuned GPT-3-(2)	**90.3**	**95**	**84.9**	**89.7**
Method	**MATH-Accuracy**	**MATH-Precision**	**MATH-Recall**	**MATH-F1**
MaxEnt	69.0	78.1	53	63.1
RefD-EQUAL	73.9	78.4	67.3	71.9
RefD-TFIDF	70.3	76.3	60.1	66.7
XGBOOST	84.1	**91**	84.3	87.5
GPT-3	63.5	89.7	30.1	45.1
Fine-tuned GPT3-(1)	71.2	89.6	79.3	81.5
Fine-tuned GPT-3-(2)	**94.4**	90.2	**95.4**	**92.7**

Table 5.10 Precision, Recall, and F-score for CS and MATH

5.5.4 Performance Comparison of Fine-Tuned GPT-3 to Other Models on CD

In the experiment with the CD, instead of fine-tuning the model from scratch, we first want to test how well fine-tuned GPT-3 does on the test instances of this new dataset without additional fine-tuning. UCD includes concepts from Computer Science and CD includes concepts from both Computer Science and Mathematics. We hypothesized that the fine-tuned GPT-3 could apply its existing knowledge to this new dataset, as the concepts in both datasets overlap in domains. We denote our model that is only fine-tuned with UCD as Fine-Tuned-GPT-3-(1). We compare the performance of our model with the other models; **MaxEnt**, **RefD-EQUAL**,

RefD-TFIDF and **XGBOOST**. To align our training and testing setup with that of other models on this dataset, we initially shuffled all the instances of MATH and CS concept pairs and randomly choose 80% instances from each domain as training and the rest of the 20% as the test data using stratified sampling. We balanced the classes in the training and test sets by oversampling the minority class which is a common technique that is applied by all of the models to which we compare our model. After the data processing, we fine-tune the Fine-Tuned-GPT-3-(1) with the CD training data and denote this model as Fine-Tuned-GPT-3-(2).

The evaluation results in Table 5.10 demonstrate that the fine-tuned GPT-3-(2) model outperforms other methods in predicting the prerequisite relations for both Mathematics (MATH) and Computer Science (CS) subjects. Specifically, the fine-tuned GPT-3-(2) model outperforms other methods in CS dataset, achieving the highest accuracy, precision, recall, and F-score. For the MATH dataset, the fine-tuned GPT-3-(2) model achieves the highest accuracy, recall, and F-score, but is outperformed by XGBOOST in precision. The MaxEnt, RefD-EQUAL and RefD-TFIDF methods show lower precision, recall, and F-score than the fine-tuned GPT-3-(2) model for both datasets. As expected, fine-tuned GPT-3-(1) is able to use some of its knowledge from UCD as it outperforms raw GPT-3 on both datasets.

5.5.5 Performance Comparison between Fine-Tuned GPT-3 and LLAMA2 on MC-PSY

We fine-tuned GPT-3 and LLAMA2 on the MC-PSY dataset, with results detailed in Table 5.11. MC-PSY contains almost equal amounts of positive and negative pairs (485 positive 515 negative). We shuffled all the pairs in the dataset and used 70% of the data as training and 30% of it as testing. The dataset is constructed by the designers of MOOCCubeX educational platform and can be found at the website Repository (2024). Before the fine-tuning, we observe that both models have a good precision but low recall, indicating a conservative approach in predicting positive instances, leading to numerous false negatives. This shows the models' initial tendency to prioritize minimizing false positives over maximizing true positives. After fine-tuning, however, both models showe improvements in recall scores without sacrificing much on precision. This shows that the fine-tuning process successfully adjusts the LLM's predictions towards a more balanced assessment of positive and negative pairs in the MC-PSY dataset. In terms of accuracy, both LLMs demonstrated comparable performance. However, fine-tuned GPT-3 beat fine-tuned

LLAMA2 in terms of F-score by a margin of 11%. This 11% margin in the F-score holds particular importance in areas like educational content creation, where accurately identifying prerequisites (true positives) is often more critical than identifying non-prerequisites (true negatives).

Model	**Recall**	**Precision**	**F1 Score**	**Accuracy**
Raw GPT-3	27.4	64.5	38.5	57.3
Fine-tuned GPT-3	88.4	60.5	71.8	66.1
Raw LLAMA2	21.2	83.8	33.9	59.7
Fine-tuned LLAMA2	51.4	73.5	60.5	67.3

Table 5.11 Performance Comparison of LLMs on MC-PSY Dataset

5.5.6 Assessment of the AI-Created Explanations

In this section, we introduce an additional experiment in which we randomly sample 100 positive pairs (A, B) from the CD and UCD test set, with our prior knowledge that fine-tuned GPT-3-(2) successfully labeled these pairs as prerequisites and generated a reasoning on why there is a prerequisite relation from A to B in these sampled pairs. Next, using our domain knowledge, we manually created our own explanations and compared them to the explanations of Fine-tuned GPT-3-(2). In order to have a fair annotation process, the 100 pairs are randomly divided into two sets, and explanations for each set are independently prepared by the authors of this paper, one PhD Computer Science student and one Computer Science Faculty Member. Before the annotation process, we decided to produce brief explanations ranging from one to five sentences. We analyzed the existing online discussions between experts and students regarding the justifications for why a concept is a prerequisite for another concept and decided to use the explanation style of the top-rated answers as a guidance. To ensure unbiased judgments, each explanation was created without access to the explanations generated by the fine-tuned GPT-3-(2) model.

The evaluation of the AI-generated explanations is centered on two main aspects. First, we control the validity of the explanations by checking if the explanation contains an incorrect information. Second, we compare the similarity of the AI-generated explanations to our manually annotated explanations and for each compared explanation, we return a similarity score between 0 and 1 where 0 represents no similarity and 1 represents perfect similarity (identical explanations).

The METEOR (Metric for Evaluation of Translation with Explicit ORdering) introduced by Lavie & Agarwal (2007) is an advanced performance metric that is primarily used for assessing the machine translation outputs against the corresponding human-generated results. METEOR is deployed in translation tasks aimed at judging the relative quality of machine translations. METEOR has shown impressive results, outperforming the widely-used BLEU metric by Papineni, Roukos, Ward & Zhu (2002), in aligning more closely with human assessments of translation quality. In our case, we use METEOR to determine how close the AI-generated explanations are to our own and report the minimum, maximum and the average METEOR score. Our custom dataset with human generated and AI generated explanations together with their METEOR scores can be found at `https://github.com/cemaytekin/GPT-3-Fine-Tuning/blob/main/explanation_list_CS.csv`. In the following Section, we present our results and discuss the weaknesses and the strengths of the model's generated explanations.

# Explanation	**# Invalid Explanation**	**Min Meteor**	**Max Meteor**	**Avg Meteor**
100	Not detected	0.135	0.67	0.39

Table 5.12 Validity and Similarity of Explanations

From Table 5.12, we can observe the performance of the explanations' validity and similarity. All 100 generated explanations are valid, as they contain no factual inaccuracies. This suggests that when the model detects a prerequisite pair, it does not make a false statement in its explanations. Although the explanations are accurate, certain differences exist between the explanations of the Fine-tuned GPT-3-(2) and ours. This occurs because a concept can be a prerequisite for another for various valid reasons. Moreover, the length and the style of the explanations can slightly vary from person to person. Despite these challenges, the correlation between the AI and human-generated explanations ranged from 0.12 to 0.67, with an average score of 0.38. While the AI-explanations did not contain invalid statements, some of the explanations were rather vague. For instance, when the model asked with a pair(Geometry, Computer Vision), instead of precisely explaining which subjects of geometry knowledge is utilized in computer vision course, it rather stated that "*Yes; understanding of geometry is crucial for computer vision algorithms*". Conversely, for some examples it provided very detailed explanations such as for the pair (Differentiable Manifolds, Differential Geometry), it stated that "*Yes; differentiable manifolds provide the basic framework for differential geometry as they allow for the definition of concepts such as vector fields, curves, and surfaces in a smooth and consistent manner. Additionally, the concepts of differential geometry are applied to differentiable manifolds to study the properties of these objects in greater*

detail. Therefore, a basic understanding of differentiable manifolds is necessary for a thorough understanding of differential geometry". We observed that the granularity of the concepts influenced the description quality. When the model is presented with broader concepts such as *Probability Theory* or *Geometry*, it produced more generalized answers regarding their prerequisite relationships. Conversely, when it encountered more narrowly defined concepts such as *Ordinary Differential Equations* or *Differentiable Manifolds*, it provided more detailed reasoning on their prerequisite relation. This demonstrates that the model skillfully adjusts its reasoning based on the specificity of the provided concepts. The highest METEOR score (0.669) is recorded for the pair (Computer Programming, Variable) where the model gave an explanation as "*Yes; programming language is a prerequisite for variable because variables are used to store data and manipulate it in different ways, which requires a programming language to be used. Additionally, programming language provides syntax and structure to create and manage variables. Therefore, having an understanding of programming language is necessary to understand the concept of variables*" which is in line with the expert explanation *Yes; knowledge of a programming language is necessary to understand and use variables as they label and store the data in the memory which can then be used by the program at any time.* The lowest METEOR score (0.135) is recorded for the pair (Information Retrieval, Algorithms) where the model gave a general explanation as "*Yes; algorithms are necessary for information retrieval as they provide the tools needed to efficiently organize, sort, and compare information*" while the expert detailed the more specific usages of *Algorithms* in *Information Retrieval* as " *Yes; IR algorithms mainly compare the query to the collection of sources with a scoring algorithm. This scoring algorithm is used to compute the relevance between the query and the document itself*".

5.5.7 Analysis of Parameter Configuration for GPT-3

In this subsection, we explore how various parameter configurations affect GPT-3's fine-tuning process. In order to do that, we fine-tune a raw GPT-3 model from scratch using different parameter values at each experiment and then evaluate the model on a test dataset. Each fine-tuning experiment utilizes an identical set of 200 pairs for a fair comparison. The pairs for the training set are chosen from UCD and CD dataset using stratified sampling. The test set consists of 100 pairs and as in the case with the training set, each experiment is validated using the same test set. The pairs in the test set, also utilized in the experiment of Section 5.5.6, consist of those for which we have manually prepared explanations illustrating why one

concept is the prerequisite of the other. Thus, for models fine-tuned with varying parameters, we assess whether they can accurately identify prerequisite relations and provide meaningful justifications. Hence, we employ the METEOR metric to simultaneously evaluate the accuracy of prerequisite detection and the quality of the explanations. METEOR scores close to 0 indicate that the prerequisite relations are not captured by the model and scores close to 1 indicate that the prerequisite relations are captured and justified with high-quality explanations similar to those annotated by humans.

Epoch	Avg Score	Min Score	Max Score	Variance of Scores
1	0.24692	0.04983	0.54504	0.00968
4	0.27003	0.13856	0.57151	0.00819
8	0.27153	0.06188	0.53126	0.00831
16	0.26963	0.05833	0.53435	0.00826

Table 5.13 METEOR score statistics for various epochs.

Table 5.13 presents four fine-tuned models, each trained with a varying number of epochs. As the number of epochs increases from 1 to 4, there is an increase in the average METEOR score from 0.25 to 0.27. However, from epoch 4 to 8, while there is a slight increase in the average score, it is not as significant. This could potentially suggest that increasing the number of epochs beyond a certain point has diminishing returns in terms of model performance. When the epochs are further increased to 16, the average METEOR score slightly decreases. This decline may indicate overfitting, suggesting the model excessively adapts to the training data and loses its generalization capabilities on unseen data. Moreover, the variance of the scores decreases from the 1st epoch to the 4th and then stays relatively consistent, indicating that the model predictions become more stable with additional epochs. The recommended number of epochs in the official documentation of GPT-3 is 4. We conclude that our results are consistent with the official GPT-3 documentation's recommendation on the selection of the number of epochs parameter.

Learning Rate	Average Score	Min Score	Max Score	Variance of Scores
0.4	0.27003	0.13856	0.57151	0.00819
0.04	0.26958	0.09433	0.48947	0.00780
4	0.27951	0.09063	0.59626	0.01135
40	0.01106	0.0	0.06880	0.00029

Table 5.14 METEOR Scores for Different Learning Rates

From Table 5.14, we observe 4 different fine-tuned models trained with varying learning rates. According to the official GPT-3 documentation, the recommended

formula for calculating the learning rate is:

$$\text{learning rate} = 0.002 \times \text{number of training examples} \times \text{learning rate multiplier}$$

Given that we used 200 examples in this training, the formula for our case becomes:

$$\text{learning rate} = 0.4 \times \text{learning rate multiplier}$$

We experimented with four learning rate multipliers (1, 0.1, 10, and 100), resulting in learning rates of 0.4, 0.04, 4, and 40, respectively. We can observe that the model with a learning rate of 4 provides the highest average score, slightly higher than the models with learning rates of 0.04 and 0.4. However, the model with a learning rate of 40 shows significantly lower performance, with its average score being much less than that of the other models. This finding suggests that an extremely high learning rate may deteriorate the model's performance. On the other hand, a moderately high learning rate like 4 may improve the performance of the fine-tuned model however, with its variance being the highest (0.01135) we may state that the model focuses too much on specific patterns in the training dataset therefore it may be sensitive to outliers or atypical instances in the test set. Additionally, the model fine-tuned with an extremely high learning rate (40) exhibits low variance but significantly underperforms, as highlighted by its average METEOR score of 0.01106. Considering both the variance and the average score, we find that using a learning rate of 0.04 or 0.4 produces better and more stable results in this setting.

5.5.8 Cost Analysis of Fine-Tuning GPT-3 and LLAMA2

In this section, we evaluate the cost of fine-tuning GPT-3 and LLAMA2 in terms of computational resources, time requirement and financial investment. The assessment is based on examples from our own experience with the fine-tuning process. Working with GPT-3 or LLAMA2 offers unique advantages. Primarily, because of their cloud-based nature, we bypass the need to use powerful local machines for computation. As of November 2023, fine-tuning a small data set with 300 examples (200 training and 100 test instances) required only about 30 seconds for GPT-3. For

the dataset with 2520 examples, the process required only about 5 minutes, highlighting a moderate increase in time despite the significantly larger dataset. The fine-tuning time for the same dataset with 2520 examples took around 1 hour to complete for LLAMA2. Financially, the cost of fine-tuning GPT-3 primarily depends on the total number of tokens used in both the training and testing sets. Although there is generally a correlation between the number of tokens and the number of words, the internal tokenizer in GPT-3 can occasionally split a single word into multiple tokens. Based on the experiments we conducted in Subsection 5.5.7, we discovered that each of our 300 fine-tuning examples averaged 63.25 words. After executing fine-tuning under various parameters eight times, we determined that the average cost for each training and testing session was approximately $2.33. This translates to a cost of approximately 12.22 cents for every 1000 words in the fine-tuning process. We should also note that doubling the number of epochs is equal to doubling the number of used tokens. In our 8 experiments, we used 45 epochs at total (1+4+8+16+4+4+4+4). Therefore, for each epoch we found the average cost to be 0.41 dollars for 18975 words of training/testing in GPT-3. For the LLAMA2 model, costs are calculated based on model usage time, with a unit cost of $0.0014 per second as of February 2024.

5.5.9 Current Limitations of Utilizing Fine-Tuned LLMs

Our study has several limitations which could be considered as future work. The first limitation is the financial and computational cost associated with fine-tuning LLMs. While fine-tuning an LLM with thousands of new data instances has a reasonable price and processing time, scalability becomes a challenge in an educational setting with millions of student-related data records. Consequently, exploring more cost-effective and efficient fine-tuning practices becomes essential. However, LLMs are in their early stages and as the new open-source models emerge, it is anticipated that cost and time associated with the fine-tuning process will diminish significantly. The second limitation is bias and fairness. LLMs can inherit biases from their training data which can lead to skewed or unfair outcomes when applied in educational settings. Therefore, careful examination of bias within these models and the development of strategies to mitigate its effects is crucial. This calls for a multidisciplinary approach, involving experts in AI ethics, education, and data science, to ensure the models operate fairly and inclusively. The third challenge is the generalization of the fine-tuned models. While the domain specific fine-tuned models are promising in terms of classification and providing valid explanatory output,

their raw, not fine-tuned versions struggle in both areas. For this reason, domain specific prerequisite detection requires fine-tuned models. With the scalability improvements in the fine-tuning process, a single fine-tuned model can be created for prerequisite detection which can operate across multiple educational domains.

6. MEKG Toolkit: A Web Application for Building and Exploring Concept Relations

6.1 Introduction

$MEKG$s provide innovative approaches for improving the teaching and learning process therefore, the main methodologies described in this dissertation are implemented in our web application [1] MEKG Toolkit. The implemented methodologies include the construction of $MEKG$s with ACE algorithm, relevant content generation using LLMs and learning path recommendations. Our web application has two main actors: students and instructors. Instructors enter the relevant concepts from a domain and then through a series of interaction with the system, they form the corresponding domain $MEKG$. On the other hand, students can utilize these constructed $MEKG$s in their study plans. If the students do not know where to start from the graph, they can demand a study order, and the system prepares a learning plan for them using the available $MEKG$s. Each learning plan includes a sequence of concepts together with their dynamic descriptions. The concepts in the sequence are ordered according to their difficulty relations which was defined as the subset of the prerequisite relation in Section 3.8. Studying from $MEKG$s offers several distinct advantages over traditional textbook-based learning. Firstly, the student knows that the presented concept will be necessary to understanding the explanations of the upcoming concepts which improves their awareness and motivation. Secondly, each description focuses on the explanation of a single concept at a time, reducing confusion and enhancing comprehension by creating a focused and streamlined learning process. Finally, through $MEKG$s, students have the capability to visualize their learning progression in real-time. This visualization is an advantage that traditional textbooks could never provide, giving students a sense of

[1] `http://aytekincem.pythonanywhere.com/`

accomplishment.

6.2 System Overview

Roles of the students and instructors are clearly defined within the entry page of the application as illustrated in Figure 6.1.

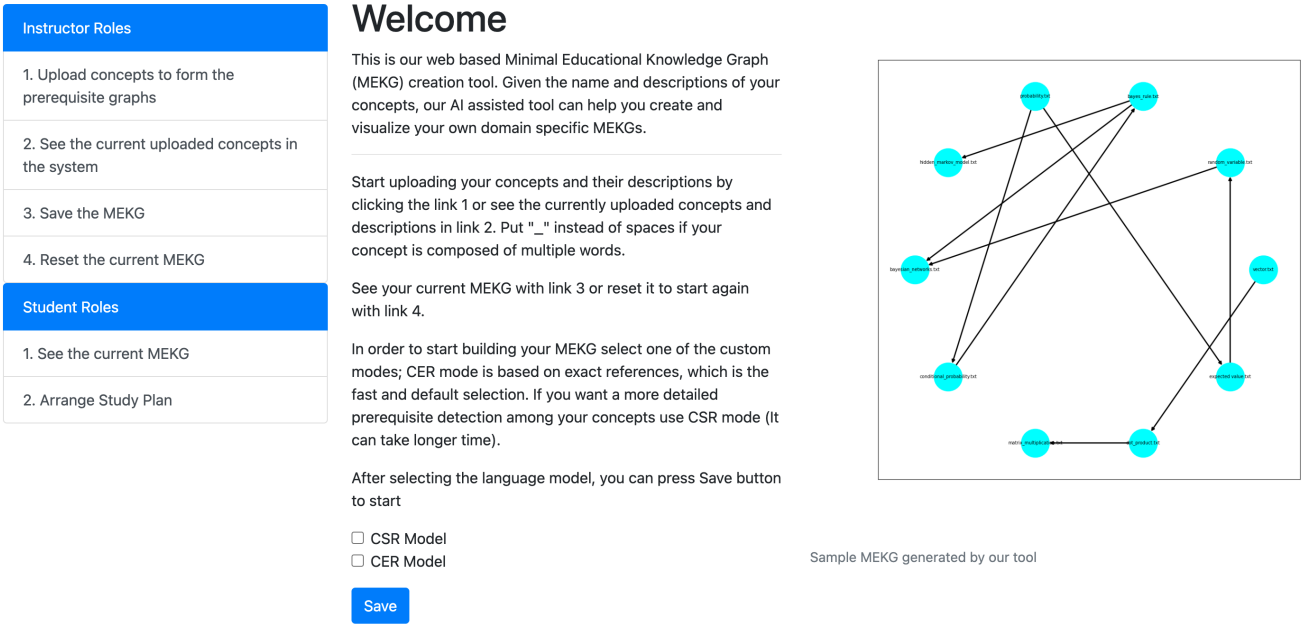

Figure 6.1 Home page of the web application

6.2.1 Instructor Role

The initial action an instructor can take is to upload a new concept to the system. By selecting the first link under “Instructor Roles”, the instructor is directed to an upload page as depicted in Figure 6.2. Here, they are prompted to provide a name and description for the concept. Instructors have the flexibility to manually enter a description or, alternatively, they can opt to generate an automatic description using an LLM. After the concept and its description are finalized, the instructor should click the “Save” button to store the selection. If the concepts selected by the instructor involve formulas and mathematical expressions in their descriptions, these

can initially be written in plain text. The system is then capable of automatically enhancing these descriptions by properly formatting the expressions and formulas. Instructor can then go back to the home page and click the second instructor link to see all the concepts present in the system.

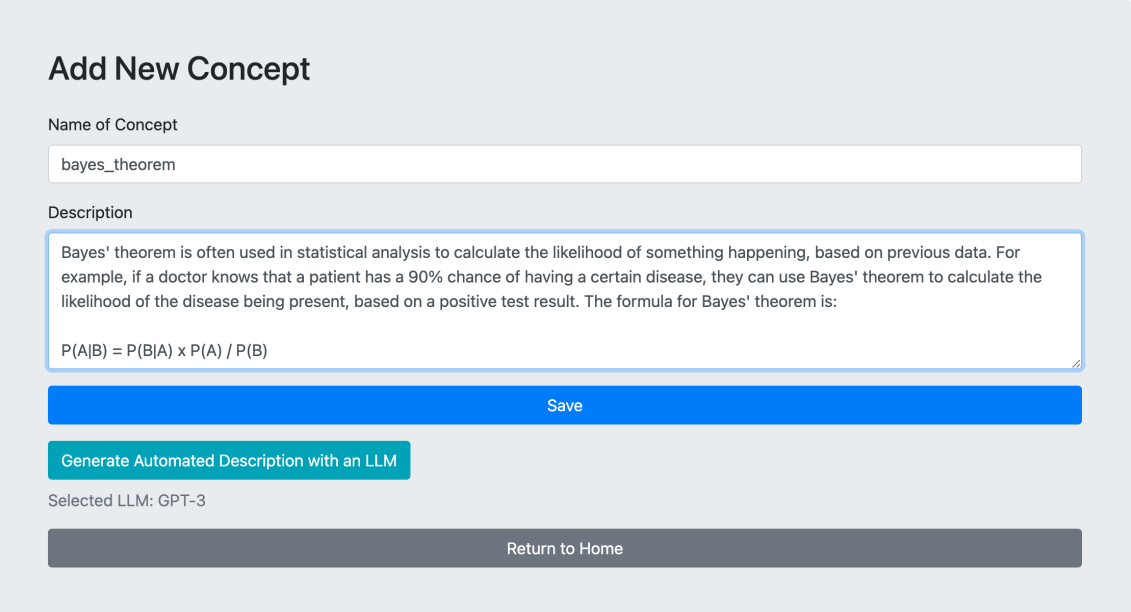

Figure 6.2 Upload page

.

After the instructor enters sufficient amount of concepts, she can request the run the ACE algorithm to build the corresponding $MEKG$. ACE can be run in two different modes: CER and CSR. CSR mode is more accurate than CER however it may run slower when the size of the considered concepts increase due to the high cost of computational cost discussed in Section 3.9. There are two checkboxes in the home page, one is called "CSR Model" and the other one is "CER Model". Upon the selection of the checkbox "CSR Model", the system is instructed to use the embedded language model all-MiniLM-L6-v2 to calculate the CSR scores. Alternatively, if the "CER Model" checbox is selected, system is instructed to calculate the exact reference scores without utilizing a language model. Once the "Save" button is clicked, all the pairwise scores are computed and one by one pairs with highest scores are presented to the expert. In order to showcase a demo scenario for instructor role, we enter six concepts from Machine Learning domain. The entered concepts and their initial descriptions are in Figure 6.2. We then select the checkbox with "CSR Model" and enter the "Save" button. The page shows a load spin indicating that the CSR scores are being computed in Figure 6.4 and then the instructor is presented with the highest scored pair in Figure 6.5. Since the size of the concept set is $n = 6$, the expert needs to evaluate at most $6 * (6 + 1)/2 - 6 = 18$ prerequisite directions.

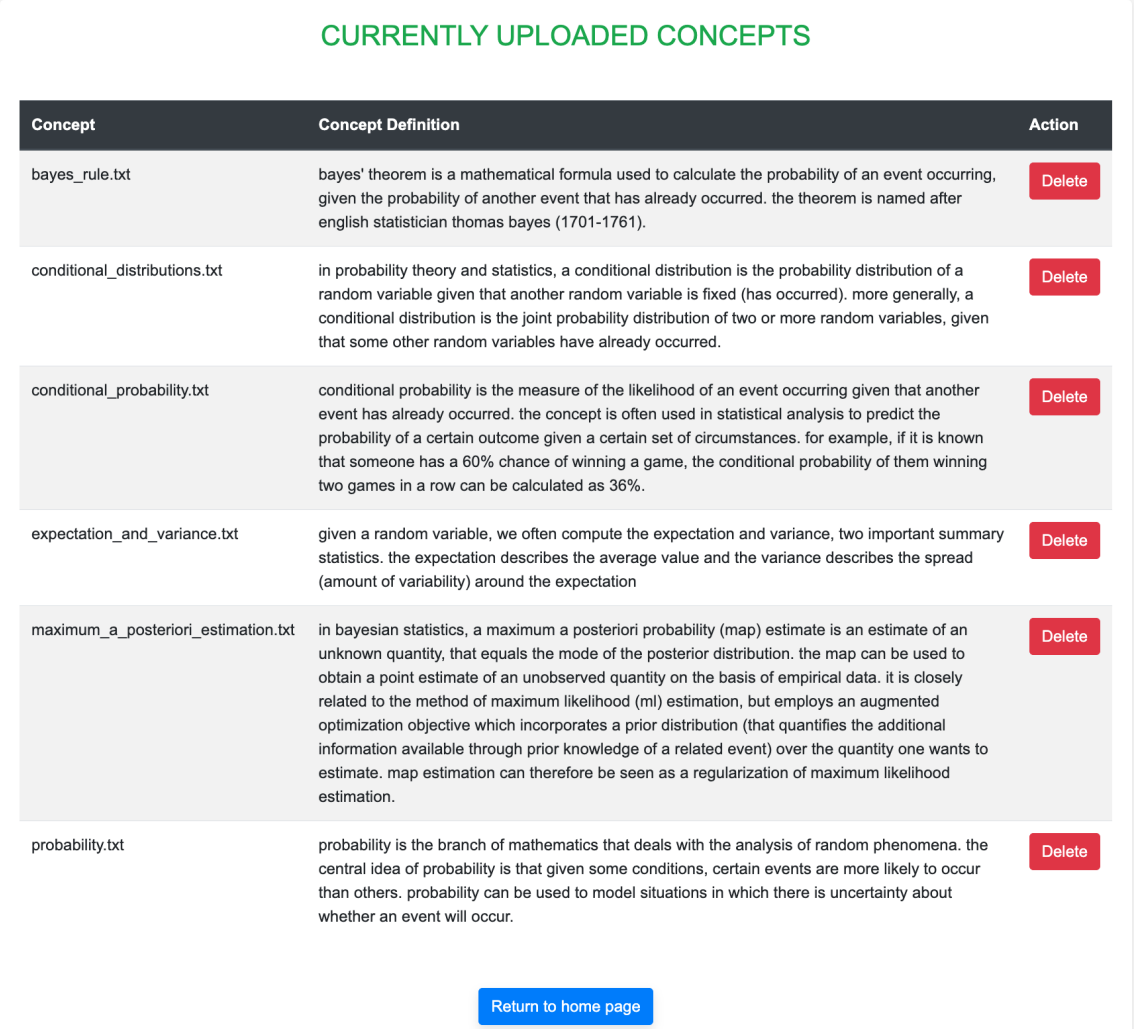

Figure 6.3 Uploaded Concepts in the System

According to the described in inference rules in Section 3.5, if one of the direction (from C_1 to C_2 or from C_2 to C_1) can already be inferred, it is not asked to the expert. In Figure 6.6, we see that there is no more concept pair left to consider. Number of questions asked so far indicate that $18 - 14 = 4$ of the pairs' label are automatically inferred. Furthermore, system also informs the instructor that the transitive edges are removed in the graph. After the instructor is done with the annotations, she can click the "Save the MEKG" link to store the graph in the system.

.

.

.

.

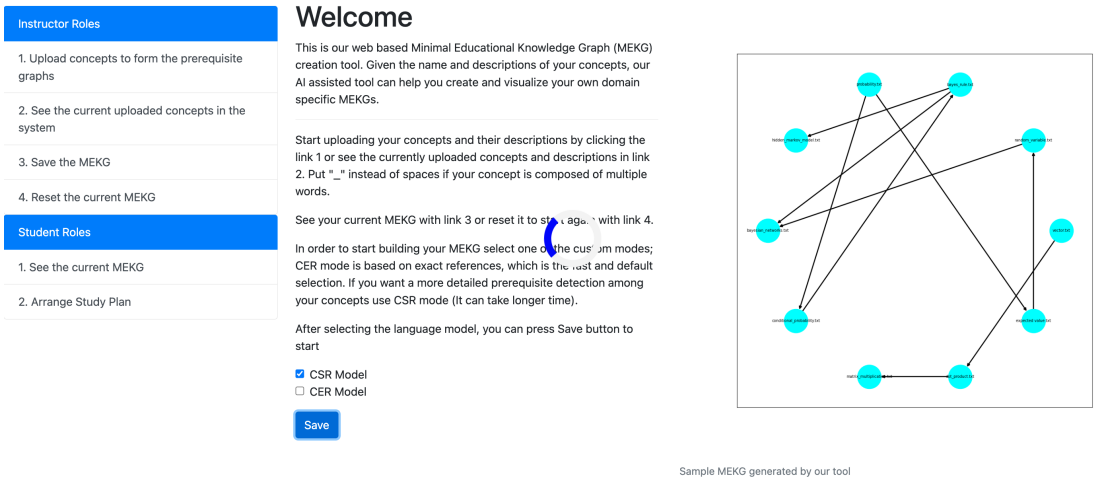

Figure 6.4 Computation of CSR scores.

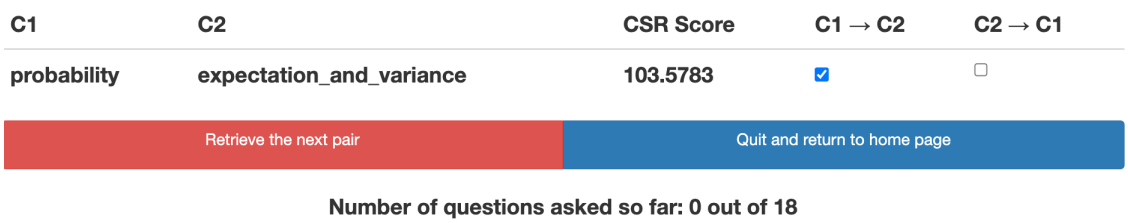

Figure 6.5 Presentation of a pair from ranked list.

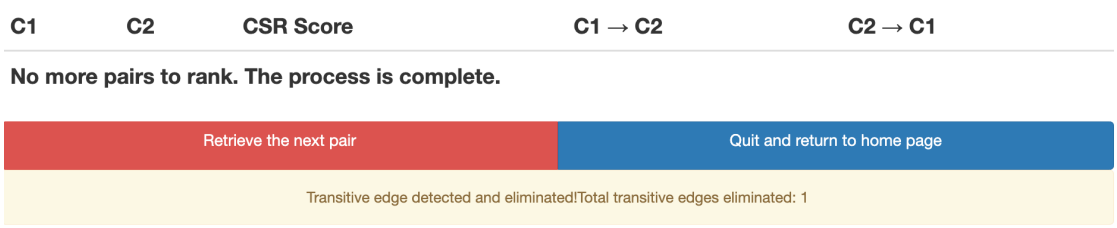

Figure 6.6 Termination of the expert interaction

.

.

.

.

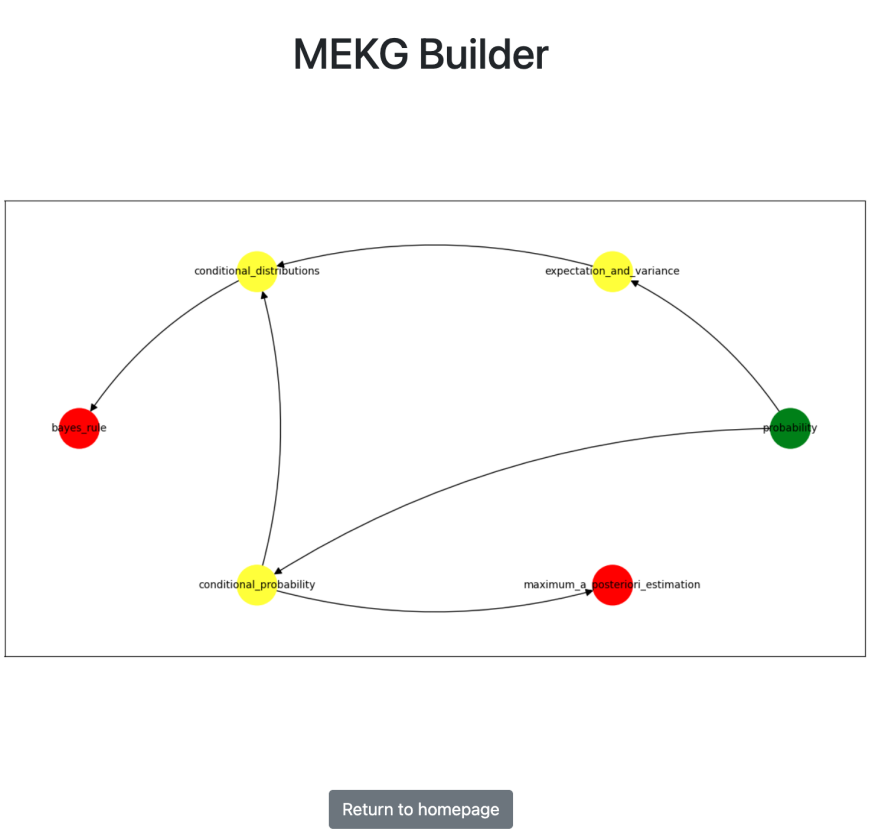

Figure 6.7 Demo MEKG for 6 concepts.

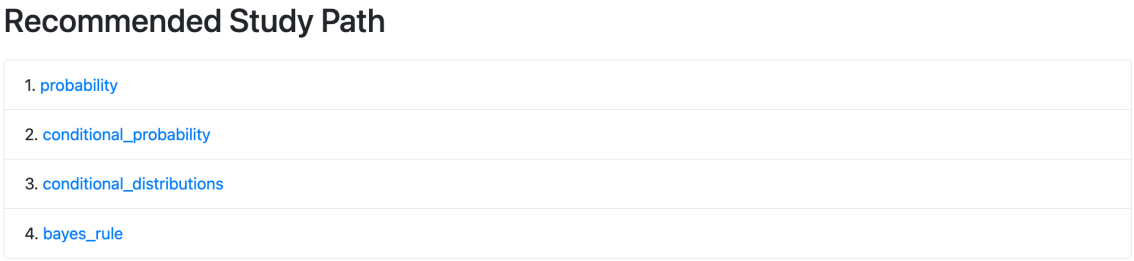

Figure 6.8 Recommended study path consisting of 4 concepts.

6.2.2 Student Role

A student entering the system can first ask to see the current $MEKG$ in the system. Continuing from our demo scenario with six concepts, when the student clicks the "See the current MEKG" link under the "Student Roles", the $MEKG$ in Figure 6.7 appears. This provides students with a comprehensive overview of the domain's concepts at first glance. If the student is interested in learning this domain, she can go back to the main page and click on the link "Arrange Study Plan". This link creates a learning path for the student by selecting a random path from the constructed $MEKG$ which consists of at least three concepts. In Figure 6.8, we see

Understanding Bayes' Theorem

Bayes' theorem is a fundamental statistical formula used to update the probability estimate of an event, based on new evidence.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Here, \(P(A|B)\) represents the conditional probability of event A given that event B has already occurred. This is calculated using the Bayes' formula below.

Explanation of Variables

In Bayes' theorem:

- $P(A|B)$ - represents the probability that event A occurs given that event B is true.
- $P(B|A)$ - represents how often event B occurs given that event A is true (often part of how we collect or understand new evidence).
- $P(A)$ - is the prior or initial probability of event A occurring before considering the evidence B.
- $P(B)$ - is the total probability of observing the evidence (event B).

Figure 6.9 An example description for Bayes' theorem

Understanding Conditional Probability

Conditional probability is the likelihood of an event occurring given that another event has already occurred. It is used to predict outcomes based on specific circumstances.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

For example, if the probability of winning a game is 60%, the conditional probability of winning two games in a row would be 36%.

Why is Conditional Probability a Prerequisite for Bayes' Theorem?

Ask LLM

Figure 6.10 An example description for Conditional probability without the justification text from LLM.

Understanding Conditional Probability

Conditional probability is the likelihood of an event occurring given that another event has already occurred. It is used to predict outcomes based on specific circumstances.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

For example, if the probability of winning a game is 60%, the conditional probability of winning two games in a row would be 36%.

Why is Conditional Probability a Prerequisite for Bayes' Theorem?

Ask LLM

Conditional probability is crucial for understanding Bayes' Theorem because it directly informs how we update our beliefs about the likelihood of events based on new information. Bayes' Theorem tweaks our initial assumptions (or probabilities) about events by incorporating the likelihood of new or additional evidence that is related to these events. Essentially, conditional probability allows us to revise the probability of an event A happening, given that another event B has occurred. Bayes' Theorem applies this in a practical way, helping us to refine our predictions or assumptions based on the interdependence of these events. This ability to update and refine probabilities using observed or new data makes conditional probability a foundational element for Bayes' Theorem, widely used in statistics, data science, and many forms of decision-making analysis where outcomes are uncertain but interconnected.

Figure 6.11 An example description for Conditional probability with the justification text from LLM.

a learning path with concepts "probability", "conditional probability", "conditional distributions" and "bayes rule". Student also has an option to request a new learning

path and in that case, a new random learning path from the $MEKG$ is created and replaced with the existing one. Each concept in the learning path is a link and the student can click on it to see its descriptions. In Figure 6.9, we see a generated description for the concept Bayes' theorem which is the last concept in the learning path and colored as red node in the $MEKG$. In the descriptions of the yellow and green colored nodes, student has also an option to request a justification text on why the concept description she is currently reading is a prerequisite for the red colored node in the end of the learning path. This is achieved through an interface which communicates with the fine-tuned LLM. In Figure 6.10, we see such a description. Upon clicking the button "Ask LLM", the question is sent as a query to the fine-tuned LLM and the return answer is displayed in the box as shown in Figure 6.11.

6.3 Implementation Details

The web application is built using Flask, a lightweight and modular web framework written in the Python programming language. Flask is chosen for its simplicity and flexibility, which facilitates rapid development and ease of use. To manage and manipulate graph data structures, we utilized NetworkX, a Python package specifically designed for the creation, manipulation, and study of complex networks and graphs. The graphs generated and utilized within the application are stored in a standard text-based format using JSON (JavaScript Object Notation), which ensures that the data remains easily readable and accessible. To incorporate the LLMs into the application, we have subscribed to OpenAI, enabling us to utilize their well-documented Application Programming Interface (API). Through this API, the application communicates with both the general and fine-tuned versions of GPT-3. For the deployment of the web application, we have opted to use PythonAnywhere, a web hosting service tailored for Python applications. The application, accessible via the domain name "aytekincem.pythonanyanywhere.com", is hosted on this platform to take advantage of its support for Python-centric environments.

7. CONCLUSION & FUTURE WORK

Throughout this work, we have explored innovative methodologies for constructing and utilizing Minimal Educational Knowledge Graphs (MEKGs) as well as investigating the automatic detection of prerequisite relations between educational concepts using fine-tuned large language models (LLMs). In Part I, we introduced an AI-assisted methodology designed to generate MEKGs that demonstrate prerequisite relations among domain-specific concepts, significantly reducing the experts' workload. Our novel prerequisite scoring algorithm, CSR, leverages semantic similarity to assign unique scores to concept pairs. We implemented this methodology in a web application, providing a comprehensive framework for experts to create MEKGs which e-learning systems can employ for tasks such as optimizing learning paths and identifying concept difficulty.

In Part II, we presented an approach that fine-tunes LLMs to automatically detect and explain prerequisite relations. We evaluated their performance across three different datasets. The LLMs demonstrated robust classification capabilities and offered credible explanations for their predictions. Our methodology underscores the adaptability of fine-tuning strategies for intelligent education systems. For future research, we aim to integrate additional relations into EKGs, such as 'subclass' relations that represent hierarchical structures between concepts, potentially revealing new rules and knowledge. Further, we hope that the released benchmark datasets and fine-tuning data will continue to advance the development and use of MEKGs and fine-tuned LLMs in intelligent educational systems, leading to more advanced tutorial technologies.

Overall, we provide a dual framework for the innovative organization of domain knowledge through MEKGs and the enhancement of intelligent education systems via fine-tuned LLMs. We hope that our contributions in this dissertation will inspire further research and lead to improved educational systems.

BIBLIOGRAPHY

Aguiar, C. Z., Cury, D., & Zouaq, A. (2016). Automatic construction of concept maps from texts, 1–6.

Barker, K. & Cornacchia, N. (2000). Using noun phrase heads to extract document keyphrases. In Hamilton, H. J. (Ed.), *Proceedings of the Thirteenth Canadian Conference on Artificial Intelligence*, volume 1822 of *Lecture Notes in Computer Science*, (pp. 40–52)., London. Springer-Verlag.

Bender, E., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? (pp. 610–623).

Chen, P., Lu, Y., Zheng, V. W., Chen, X., & Yang, B. (2018). Knowedu: A system to construct knowledge graph for education. *IEEE Access*, *6*, 31553–31563.

Chiou, C.-C., Tien, L.-C., & Lee, L.-T. (2015). Effects on learning of multimedia animation combined with multidimensional concept maps. *Computers & Education*, *80*, 211–223.

Corpus (2024). Corecorpus: Online database.

Cui, H., Lu, J., Wang, S., Xu, R., Ma, W., Yu, S., Yu, Y., Kan, X., Ling, C., Ho, J., et al. (2023). A survey on knowledge graphs for healthcare: Resources, applications, and promises. *arXiv preprint arXiv:2306.04802*.

DAG (2024). Binary relations - explanation and examples.

Dang, F.-R., Tang, J.-T., Pang, K.-Y., Wang, T., Li, S.-S., & Li, X. (2021). Constructing an educational knowledge graph with concepts linked to wikipedia. *Journal of Computer Science and Technology*, *36*(5), 1200–1211.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Dictionary, C. (2024).

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., & Hon, H.-W. (2019). Unified language model pre-training for natural language understanding and generation. In *Neural Information Processing Systems*.

Doshi-Velez, F. & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Fettach, Y., Ghogho, M., & Benatallah, B. (2022). Knowledge graphs in education and employability: A survey on applications and techniques. *IEEE Access*, *10*, 80174–80183.

Gong, W., Smith, D., Wang, Z., Barton, C., Woodhead, S., Pawlowski, N., Jennings,

J., & Zhang, C. (2022). Neurips competition instructions and guide: Causal insights for learning paths in education.

Google Knowledge Graphs (2012). Google Knowledge Graphs. Accessed: 2022.

Hirashima, T., Yamasaki, K., Fukuda, H., & Funaoi, H. (2015). Framework of kit-build concept map for automatic diagnosis and its preliminary use. *Research and Practice in Technology Enhanced Learning*, *10*(1), 1–21.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. d., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., et al. (2021). Knowledge graphs. *ACM Computing Surveys (CSUR)*, *54*(4), 1–37.

IBM Knowledge Graphs (2024). Ibm knowledge graphs. Accessed: 2024.

Imamguluyev, R. (2023). The rise of gpt-3: Implications for natural language processing and beyond. *International Journal of Research Publication and Reviews*, *4*, 4893–4903.

Jia, C., Shen, Y., Tang, Y., Sun, L., & Lu, W. (2021). Heterogeneous graph neural networks for concept prerequisite relation learning in educational data. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., & Zhou, Y. (Eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (pp. 2036–2047)., Online. Association for Computational Linguistics.

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M., Weller, J., Kuhn, J., & Kasneci, G. (2023). Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, *103*, 102274.

Khosravi, H., Denny, P., Moore, S., & Stamper, J. (2023). Learnersourcing in the age of ai: Student, educator and machine partnerships for content creation. *Computers and Education: Artificial Intelligence*, *5*, 100151.

Lavie, A. & Agarwal, A. (2007). Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments, 228–231.

Lee, S., Park, Y., & Yoon, W. C. (2015). Burst analysis for automatic concept map creation with a single document. *Expert systems with applications*, *42*(22), 8817–8829.

Li, Z., Cheng, L., Zhang, C., Zhu, X., & Zhao, H. (2023). Multi-source education knowledge graph construction and fusion for college curricula. In *2023 IEEE International Conference on Advanced Learning Technologies (ICALT)*, (pp. 359–363). IEEE.

Liang, C., Wu, Z., Huang, W., & Giles, C. L. (2015). Measuring prerequisite relations

among concepts. In *Conference on Empirical Methods in Natural Language Processing*.

Liang, C., Ye, J., Wang, S., Pursel, B., & Giles, C. L. (2018). Investigating active learning for concept prerequisite learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Liang, C., Ye, J., Wu, Z., Pursel, B., & Giles, C. L. (2017). Recovering concept prerequisite relations from university course dependencies. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*.

Liang, C., Ye, J., Zhao, H., Pursel, B., & Giles, C. L. (2019). Active learning of strict partial orders: A case study on concept prerequisite relations. *arXiv preprint arXiv:1801.06481*.

Library, G. (2024). gensim.

Manrique, R., Pereira, B., & Mariño, O. (2019). Exploring knowledge graphs for the identification of concept prerequisites. *Smart Learning Environments*, *6*(1), 1–18.

Metacademy (2024). Metacademy. Accessed: 2022.

Microsoft (2023). Microsoft.

Mihalcea, R. & Tarau, P. (2004). TextRank: Bringing order into text. In Lin, D. & Wu, D. (Eds.), *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, (pp. 404–411)., Barcelona, Spain. Association for Computational Linguistics.

MiniLM (2023). all-MiniLM-L6-v2.

Molontay, R., Horváth, N., Bergmann, J., Szekrényes, D., & Szabó, M. (2020). Characterizing curriculum prerequisite networks by a student flow approach. *IEEE Transactions on Learning Technologies*, *13*(3), 491–501.

MOOC report (2024). MOOC report.

Novak, J. D. (2010). *Learning, creating, and using knowledge: Concept maps as facilitative tools in schools and corporations*. New York, NY, USA: Routledge.

Novak, J. D. & Cañas, A. J. (2006). The theory underlying concept maps and how to construct them. *Florida Institute for Human and Machine Cognition*, *1*, 2006–2001.

OpenAI (2023). Gpt-3 parameters. Accessed: 2023.

OpenAI (2024). Introducing gpt-3.

Pan, L., Li, C., Li, J., & Tang, J. (2017). Prerequisite relation learning for concepts in MOOCs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 1447–1456).,

Vancouver, Canada. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). Bleu: a method for automatic evaluation of machine translation.

Paulheim, H. (2016). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, *8*, 489–508.

Pinandito, A., Prasetya, D. D., Hayashi, Y., & Hirashima, T. (2021). Design and development of semi-automatic concept map authoring support tool. *Research and Practice in Technology Enhanced Learning*, *16*(1), 1–19.

Quora (2024). Quora online discussions.

Repository, M. (2024). Mooccubex: A data repository for moocs. Accessed: 2023-10-05.

Roy, S., Madhyastha, M., Lawrence, S., & Rajan, V. (2019). Inferring concept prerequisite relations from online educational resources. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 9589–9594.

Sabanci University CS Curriculum (2023). Sabanci University CS Curriculum.

Sayyadiharikandeh, M., Gordon, J., Ambite, J.-L., & Lerman, K. (2019). Finding prerequisite relations using the wikipedia clickstream. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, (pp. 1240–1247)., New York, NY, USA. Association for Computing Machinery.

sentence transformers (2023). sentence-transformers. Accessed: 2023.

Shneiderman, B. (2020). Human-centered artificial intelligence: Three fresh ideas. *AIS Transactions on Human-Computer Interaction*.

Shokrzadeh, Z., Feizi-Derakhshi, M.-R., Balafar, M.-A., & Bagherzadeh Mohasefi, J. (2023). Knowledge graph-based recommendation system enhanced by neural collaborative filtering and knowledge graph embedding. *Ain Shams Engineering Journal*, 102263.

Singhal, A. (2012). Introducing the knowledge graph: things, not strings. 2020-11-13.

Talukdar, P. & Cohen, W. (2012). Crowdsourced comprehension: Predicting prerequisite structure in Wikipedia. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, (pp. 307–315)., Montréal, Canada. Association for Computational Linguistics.

TextRazor (2024). Textrazor - natural language processing api. Accessed: 2023-10-01.

Touvron, H., Martin, L., Stone, K., Albert, et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wikipedia contributors (2024). Named-entity recognition — Wikipedia, the free encyclopedia.

Yu, J., Wang, Y., Zhong, Q., Luo, G., Mao, Y., Sun, K., Feng, W., Xu, W.-H., Cao,

S., Zeng, K., Yao, Z., Hou, L., Lin, Y., Li, P., Zhou, J., Xu, B., Li, J.-Z., Tang, J., & Sun, M. (2021). Mooccubex: A large knowledge-centered repository for adaptive learning in moocs. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.

Zhang, J., Lan, H., Yang, X., Zhang, S., Song, W., & Peng, Z. (2022). Weakly supervised setting for learning concept prerequisite relations using multi-head attention variational graph auto-encoders. *Knowledge-Based Systems*, *247*, 108689.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., & Irving, G. (2019). Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

APPENDIX A

Missing proofs for the formula in Section 3.7.1

Proof of Proposition 1

We claim that if $G_{dir}(N, E^*)$ does not contain a transitive edge then its undirected version $G(N, E)$ is a simple triangle-free graph. We denote the maximum possible number of distinct edges among any nodes $G_{dir}N(A)$, $G_{dir}N(B)$, $G_{dir}N(C)$ as n_{max} and prove that $n_{max} = 2$. This is because if $n_{max} = 3$, there are 8 total possible edge configurations for nodes A, B and C as shown in Table A.1. Each of them leads to a transitive edge. When $n_{max} > 3$, one of the 8 possible cases for $n_{max} = 3$ is also true.

Proof by contradiction: Assume G_{dir} does not contain a transitive edge and its undirected version G is not a triangle-free graph. Then G should have at least 3 distinct edges among nodes A, B and C. However, this requires $n_{max} > 2$ which shows a contradiction.

Case	$G_{dir}E^*(A,B)$	$G_{dir}E^*(B,C)$	$G_{dir}E^*(A,C)$
1	$A \to B$	$B \to C$	$A \to C$
2	$A \to B$	$B \to C$	$C \to A$
3	$A \to B$	$C \to B$	$A \to C$
4	$A \to B$	$C \to B$	$C \to A$
5	$B \to A$	$B \to C$	$A \to C$
6	$B \to A$	$B \to C$	$C \to A$
7	$B \to A$	$C \to B$	$A \to C$
8	$B \to A$	$C \to B$	$C \to A$

Table A.1 8 possible configurations of edges among nodes A,B and C.

Proof of Proposition 2

If $G(N, E)$ is a triangle free graph then we can always form a $G_{dir}(N, E^*)$ which contains neither a cycle nor a transitive edge.

First, G is a triangle free graph, so $n_{max} = 2$. In order to have a transitive edge in G_{dir}, n_{max} should be equal or greater than 3, therefore we conclude that G_{dir} can not contain a transitive edge.

Second, since $G(N, E)$ is a simple undirected graph, all the paths in G have distinct nodes except the paths $A - n_1 - n_2 - \cdots - A$, which have the same beginning and end node. These paths are called simple cycles. Therefore, $G(N, E)$ can either contain a simple cycle or not. If $G(N, E)$ contains a simple cycle, then we can always create a non-cyclic path in $G_{dir}(N, E^*)$ by not assigning the same direction for all the edges in the path. Otherwise, if $G(N, E)$ does not contain a simple cycle, then every path in G_{dir} will inherently be non-cyclic.

Proof of Mantel's Theorem

If $G(N, E)$ is a simple undirected triangle-free graph, then its adjacent vertices have no common neighbors. So for an edge $E(x, y)$ we have $d(x) + d(y) \leq n$ where $d(x)$ shows the number of nodes x is connected to and n represents the total number of nodes in G.

The following formula

$$\sum_{x \in V} d(x)^2 = \sum_{x,y \in E} (d(x) + d(y)),$$

suggests that, if $d(x) = c$ then node x will be summed c times in the right hand side of the equation because c number of unordered edges will contain the node x. Since $d(x)$ was assumed to be c, its contribution in the summation will be $c * c$ which equals to $d(x)^2$.

If we assume G contains m edges, then we have the inequality:

$$\sum_{x \in V(G)} d(x)^2 = \sum_{xy \in E(G)} (d(x) + d(y)) \leq n \times m.$$

Moreover, the Cauchy-Schwarz inequality states that for any sequences of real numbers $\{a_i\}$ and $\{b_i\}$:

$$\left(\sum_i a_i b_i\right)^2 \leq \left(\sum_i a_i^2\right)\left(\sum_i b_i^2\right)$$

To apply this inequality in our derivation we, define $a_i = d(x_i)$ and $b_i = 1$ for all nodes $x_i \in N$.

By applying Cauchy-Schwarz:

$$\left(\sum_{x_i \in G} a_i b_i\right)^2 \leq \left(\sum_{x_i \in N} a_i^2\right)\left(\sum_{x_i \in N} b_i^2\right)$$

Substituting $a_i = d(x_i)$ and $b_i = 1$, the inequality becomes:

$$\left(\sum_{x \in N} d(x) \cdot 1\right)^2 \leq \left(\sum_{x \in N)} d(x)^2\right)\left(\sum_{x \in N} 1\right)$$

Simplifying further:

$$\left(\sum_{x \in N} d(x)\right)^2 \leq \left(\sum_{x \in N} d(x)^2\right) \cdot n$$

we have:

$$\frac{1}{n}\left(\sum_{x \in N} d(x)\right)^2 \leq \sum_{x \in N} d(x)^2.$$

Furthermore, by the Handshaking Lemma we know that,

$$\sum_{x \in N} d(x) = 2m$$

Therefore,

$$\frac{1}{n}(2m)^2 \leq nm$$

which simplifies to:

$$\frac{1}{n} \cdot 4m^2 \leq nm$$

and further simplifying:

$$\frac{4m^2}{n} \leq nm$$

Dividing both sides by m:

$$\frac{4m}{n} \leq n$$

Finally, multiplying both sides by n and dividing by 4 yields:

$$m \leq \frac{n^2}{4}$$