Doctoral Thesis

Studies on Content Analysis and Ordering of Courses from a Knowledge-Based Perspective

Yiling Dai

February 2021

Department of Social Informatics
Graduate School of Informatics
Kyoto University

Doctoral Thesis
submitted to Department of Social Informatics,
Graduate School of Informatics,
Kyoto University
in partial fulfillment of the requirements for the degree of
DOCTOR of INFORMATICS

Thesis Committee:	Masatoshi Yoshikawa, Professor
	Keishi Tajima, Professor
	Hiroaki Ogata, Professor

Studies on Content Analysis and Ordering of Courses from a Knowledge-Based Perspective*

Yiling Dai

Abstract

With the development of open and online education, students have easier access to a larger amount of learning materials. One of such resources, Massive Open Online Courses (MOOCs) make it possible for students to select courses based on their own learning goals. On the other hand, the one-fits-all curriculum policy in traditional higher education cannot prepare students for the competitive and segmented job market. To this end, adapting learning experience according to the rapidly-changing job market is essential to achieve fruitful learning and successful career development. We envision a picture of higher education in which the students are encouraged to select courses and design a personalized curriculum for various learning goals. Facilitating the integration and adoption of online courses motivates this thesis.

With diversified learning goals and heterogeneous courses, it is challenging for students to discover appropriate courses. In this thesis, we tackle the problems of course content analysis and course ordering from a knowledge-based perspective. In other words, we put emphasis on the matching and analysis of learning goals and courses at the level of knowledge. Considering different types of learning goals and scales of courses, we address the following three tasks:

- *Course content modeling*. This task models the knowledge taught in a single course, helping understand and evaluate the course content. A curriculum guideline is utilized as the domain knowledge categorization to indicate what categories of knowledge are contained in the course. Specifically, we propose a Wikipedia structure-based method to capture the implicit relatedness of a course syllabus and the descriptions of the domain knowledge categories.

*Doctoral Thesis, Department of Social Informatics, Graduate School of Informatics, Kyoto University, KU-I-DT6960-27-1468, February 2021.

- *Knowledge coverage estimation*. In this task, we assume that the student has a specific learning goal, and estimate how much of the target knowledge is covered in a single course. We define a concept of knowledge category coverage as the proportion of the knowledge taught in the course to the knowledge required in the category. We then model the knowledge category and the course as a set of concepts, and utilize the centrality upon a taxonomy to quantify the importance of concepts for the computation of the coverage.

- *Course ordering*. This task identifies the optimal order to take related courses for a given learning goal. We focus on "technical terminologies" which are frequently required in the job market. Given a technical terminology, we aim at identifying an order of courses which contributes to the acquisition of the terminology and also follows the prerequisite relationships among courses. We develop a two-step approach in which course-terminology relatedness is firstly estimated and then courses are ordered based on the prerequisite relationships and the estimated relatedness. In addition to an information retrieval-oriented evaluation metric, we explore whether the order is effective from pedagogical perspectives.

This thesis has advantages over existing works as it a) supports direct learning goals represented as domain knowledge categories and job opportunities, b) provides a systematic view on course content by utilizing the domain knowledge categorization, c) quantifies knowledge coverage to provide a helpful criterion in course selection, and d) models a rank-sensitive ordering problem where the information gain of every position is optimized. Furthermore, promising results were obtained to show that the proposed methods are effective.

Keywords: Course content analysis, Knowledge coverage, Course ordering, Curriculum guideline, Job opportunity

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

In this chapter, we first introduce the background and motivation of this thesis. Then, we state the challenges and research tasks. After that, a differentiation with related works follow. At last, we outline the thesis structure.

1.1 Background and Motivation

1.1.1 Cutting across the institutional boundaries

As the printing press changed the spread of textbooks, and the television brought the Open University into being, the Internet is reshaping higher education [1]. Among various educational innovations which are powered by information technologies, Massive Open Online Courses (MOOCs) have earned growing popularity and triggered discussions since 2012. Rather than merely digitalizing the course materials or transiting the in-class activities into a closed online learning system, MOOCs aim at providing high-quality education to anyone with Internet access.

Here, we highlight a core characteristic of the MOOC environment— the courses and students are large at scale and rich in diversity.

- **Courses**. As one of the biggest MOOC platforms Edx reports, it has built partnerships with more than 140 institutions and succeeded in providing more than 2,500 courses.[1] These courses cover across various subjects including computer science, math, chemistry, business, humanities, and so on.

- **Students**. Belanger and Thornton [2] reported that their first MOOC reached around 12,000 students, more than half of whom actually interacted with the course materials. Although only 313 students completed the course successfully, it is noteworthy that those students represented at least 37 different countries.

The massiveness and openness make MOOC a learning environment with considerably more freedom than traditional higher education. Instead of completing the prescribed curriculum and achieving a degree from the university, MOOC students demonstrate a wide range of motivations. Acquiring knowledge of the subject, updating skills in the current job position, gaining opportunities for career changes, and having fun by learning new things are some frequently observed motivations to take MOOCs [3]. In other words, the students demonstrate more diversified learning goals when they have access to a larger amount of learning materials. Therefore, it is necessary to discover the courses that meet various learning goals.

1.1.2 Bridging the gap between academia and industries

The gap between academia and industries has been a long-standing problem. Taking the domain of computer science for example, many research works have investigated what knowledge is perceived important by different parties such as educators, graduates, workers, and recruiters [4, 5, 6, 7, 8, 9]. Here, we highlight some interesting findings. Lethbridge's survey [5] found that the knowledge taught in a computer science program, such as digital electronics and formal languages, are not actually applied in the software professionals' work, thus, tends to be forgotten after students' graduation. For specific software (e.g., educational software) designers, they reflected that knowledge about user experience and interface design was extremely important in their jobs but under-taught in their formal education [7]. Exter et al. [8] reported that acquiring an understanding of the underlying concepts of programming languages was perceived much more beneficial than learning the most recent programming languages. To this end, a predefined curriculum cannot serve the ever-changing and segmented job requirements.

To bridge the gap between academia offerings and industrial demands, activities which combine learning and working were suggested to be integrated into the curric-

[1] `https://www.edx.org/`, accessed November 25, 2020.

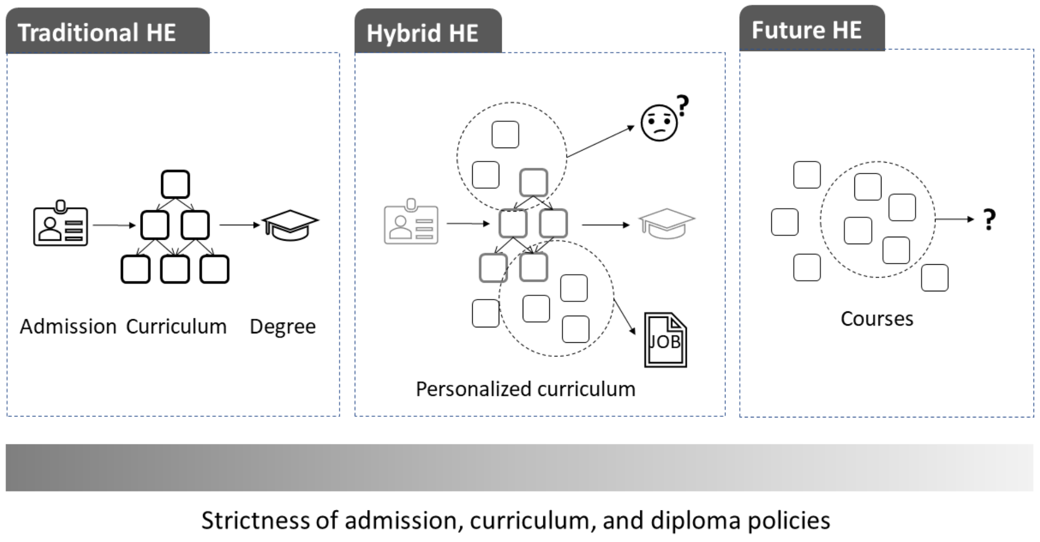


Figure 1.1: Envisioning higher education.

ular [7, 10]. Such activities could be internships, realistic and complex projects, or even special talks given by industrial professionals. These activities were expected to strengthen students' motivations to pursue their majors in the future [11] and demonstrated a relation with the improvement of students' engagement in learning [12]. To sum up, there is a need to modify the students' learning experiences according to their career demands at flexible time points.

1.1.3 Envisioning future higher education

As described in Sections 1.1.1 and 1.1.2, we envision a picture of higher education in which the "spatial walls" between educational institutions and the "temporal walls" between learning and working are lowered or even disappear. In Figure 1.1, we map possible types of higher education into a spectrum according to the degree of strictness of the admission, curriculum, and diploma policies. As you can see, the traditional higher education locates at the left extreme of the spectrum. With a strictly prescribed curriculum, the students are deprived of the opportunities to consider what to learn by themselves. At the other extreme of this spectrum, we imagine a completely free environment where the students can decide whatever courses to take whenever necessary.

Actually, it is still under debate whether such a free environment of higher education can empower the society better and how many endeavors and collaborations are necessary to achieve such a revolution. However, we do observe the higher education shifting to the right extreme gradually. Initiatives of using and adopting open courseware and MOOCs at the institutional or governmental level have been established across the world, including Europe [13], Arab [14], India [15], China [16], and so on. Although the development of open and online education is at different stages in these regions, a commonly underlined motivation is to create accessible and flexible learning opportunities for the students. As a result, we consider the higher education in the coming decades as a hybrid one in which open learning materials (e.g., MOOCs) are integrated on demand into the students' formal curriculum, to fulfill their personal or career development. Facilitating the integration and adoption of online courses motivates this thesis.

1.2 Challenges and Tasks

In a hybrid environment in which the formal learning in traditional institutions and the personalized learning on the Internet mix, many stakeholders including students, educators, recruiters, and policy makers will face the following challenges:

Challenge 1 Expanded and diversified learning goals.

In addition to the completion of a prescribed curriculum, the students take courses to enrich their knowledge or prepare for a future career. Commonly, the curricula in traditional higher education institutions put more emphasis on the theoretical construction of the knowledge in the domain, while job-oriented programs focus on how the knowledge is applied in the real world. Besides, for the learning goal such as personal interests towards a subject, the goal is inherently unclear and difficult to be converted into a specific set of learning outcomes. We suppose that students have these learning goals contextually, though with different frequencies. Therefore, it is helpful to understand how the courses serve different types of learning goals. Furthermore, it is important to encourage the students to explore and learn beyond their original learning goals.

Challenge 2 A large and heterogeneous collection of courses.

It is beneficial that the students have more choices of courses provided by different institutions. However, these courses are designed under different educational purposes and have different focuses even on a same topic, which makes it difficult to identify the contents of the courses. Understanding how the courses are similar or different to others and how they could be combined is important in planning a successful learning.

Facing the above challenges, this thesis addresses the content analysis and the ordering of courses from a knowledge-based perspective. A knowledge-based perspective means that we discuss both the courses and the learning goals at the level of knowledge. In other words, we put emphasis on the intrinsic aspects of the courses rather than extrinsic aspects, such as the popularity of the courses, the reviews of the courses, the media used in the courses, and the in-class activities conducted in the courses. The whole course materials such as lecture notes, slides, and video clips are considered as rich resources to represent the knowledge being taught. However, they also contain extrinsic aspects indicating how the knowledge is conveyed, such as the layout of the slides and the use of images and figures. Separating the intrinsic and extrinsic aspects of the courses is an orthogonal research area and beyond the scope of this thesis. Instead, we utilize the course syllabus as a cleaner information source to capture the knowledge that is taught in the courses.

More specifically, we approach by considering different types of learning goals and different scales of courses as follows:

- **Non-specific and specific learning goals**. In some situations, the students have a vague interest towards a subject or just want to explore the content of world-class courses. We name this type of learning goals as non-specific and understanding the course content is a vital step. In contrast, the students may hold specific learning goals such as “working as a web developer in the future” and “mastering the knowledge of database at the level of a university graduate”. For such learning goals, identifying the courses that teaches the target knowledge is our major interest.

- **A single course and multiple courses**. When examining on a single course, we aim at understanding what knowledge and how much of it is taught in the course. For multiple courses, we need to take into account more aspects other than the knowledge contained in individual courses. For example, we need to answer questions including but not limited to: Which course is preferable of these two similar courses? What combination of courses is the most efficient way to acquire the target knowledge? In what order should I take the courses?

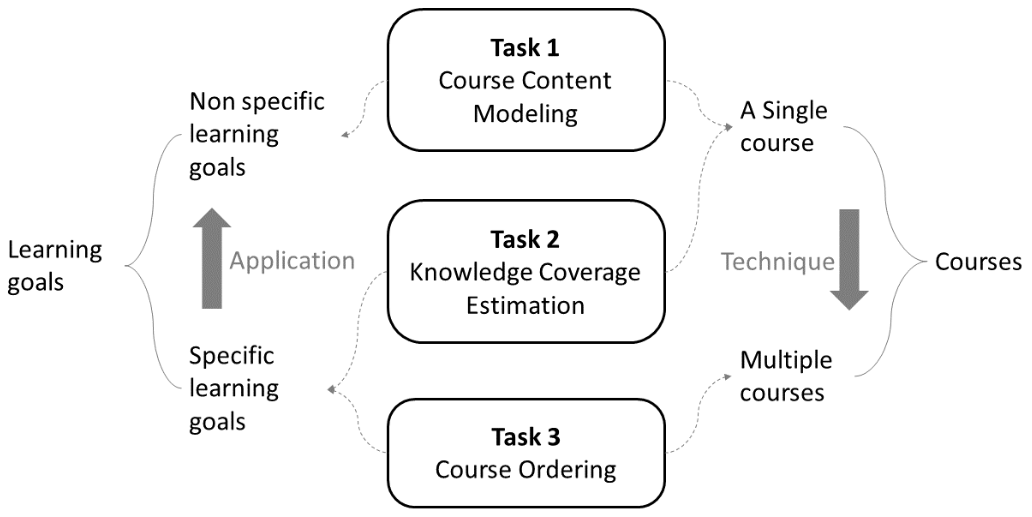


Figure 1.2: Thesis framework.

We then define three essential tasks from the above dimensions as shown in Figure 1.2:

Task 1 *Course content modeling*. This task models the knowledge taught in the course by projecting it to a predefined knowledge categorization, which helps identify, understand, and evaluate the content of a single course.

Task 2 *Knowledge coverage estimation*. This task assumes that the student has a specific learning goal and estimates how much of the target knowledge is covered in a single course. The estimated knowledge coverage can be used as a straightforward criterion to select the course that meets the goal most.

Task 3 *Course ordering*. This task identifies the optimal order to take multiple courses for a given learning goal while paying attention to the dependency relationships between courses.

There are no apparent priorities of the three tasks as they are important in different senses. From the perspective of learning goals, tasks that support specific learning goals should be prioritized as they provide direct benefit to the end users. Besides, converting non-specific learning goals into specific ones is important for making an informed learning decision. While from the perspective of courses, the techniques developed in the tasks that dealing with a single course can enhance the performance of tasks dealing with multiple courses. Besides, to avoid unnecessary pipelines in one task, we address these tasks independently in this thesis. In the following sections, we introduce the three tasks in detail.

1.2.1 Course content modeling

Modeling the content of a course in a straightforward manner is crucial in the process of choosing an appropriate course for a student. To model the course content, we address the extraction of the knowledge distribution of a course over a domain knowledge categorization in this work. A curriculum guideline is utilized as the domain knowledge categorization, which helps to cast light upon how courses are related to each other and positioned in this domain. Specifically, we propose a Wikipedia structure-based method to capture the implicit relatedness of a course syllabus and the descriptions of the domain knowledge categories.

1.2.2 Knowledge coverage estimation

In this work, we assume a situation in which the student has targeted on the knowledge defined by some category. Then, knowing how much of the knowledge in the category is covered by the courses will be helpful in the course selection. We define a concept of knowledge category coverage and estimate it in a semi-automatic manner. We first model the knowledge category and the course as a set of concepts, and then utilize a taxonomy and the idea of centrality to differentiate the importance of concepts. Finally, we obtain the coverage value by calculating the proportion of the concepts taught in the course to the ones required in a knowledge category.

1.2.3 Course ordering

As described in Section 1.1.2, obtaining job opportunities is an important purpose of learning. In this work, we focus on “technical terminologies” which are frequently required in the job market. Given a technical terminology, we aim at identifying an order of courses which contributes to the acquisition of the terminology and also follows the prerequisite relationships between courses. To solve the course ordering problem, we develop a two-step approach in which course-terminology relatedness is firstly estimated and then courses are ordered based on the prerequisite relationships and the estimated relatedness. Putting emphasis on the second step, we propose a method based on Markov decision process and compare it with three other methods. In addition to an information retrieval-oriented evaluation metric, we explore whether the order is effective from pedagogical perspectives.

1.3 Advantages over Related Work

In this section, we summarize the main advantages of this thesis over related works. Differentiation with related works in a more specified context will be introduced in the corresponding chapter of each task.

1.3.1 Supporting direct learning goals

Learning goals indicate why the students want to learn. Strong and clear goals lead to successful learning experiences. We define direct learning goals as clear statements of what the students are able to understand or do after learning, such as “understanding relational database” and “being able to design the database schema of a course enrollment system”. On the other hand, indirect learning goals are not relevant to the knowledge itself, but the secondary outcomes of learning such as the reward of succeeding or the punishment of failing the learning. “Passing the final exam of this course” counts as an example of indirect learning goals.

A bunch of previous works have attempted to recommend learning materials for indirect learning goals, such as recommending the quiz questions that are highly probable to be answered correctly by the students [17, 18, 19], recommending courses for obtaining high grades [20, 21], graduating early [20], and fulfilling implicit interest towards the courses [22, 23, 24, 25]. There exist two drawbacks in these works: a) These works greatly depend on the past learning data and assume that frequently learnt materials are preferable, which are actually not necessarily suitable to every student; b) Although indirect learning goals may improve students’ motivations, direct learning goals, in other words, the sheer curiosity towards the knowledge, are expected to trigger students’ deeper engagement in learning [26].

In this thesis, we model two types of direct learning goals: the domain knowledge categories and the knowledge required in the job market. The domain knowledge categories represent the domain knowledge in an easy-to-understand manner, which can be utilized as a reference by both expert and novice students. **Task 1** and **Task 2** aim at identifying how the courses cover the knowledge defined in the knowledge categories.

On the other hand, the job market is a major usage of the knowledge the students have learnt. To this end, using job opportunities to model the learning goals provides the students clearer images of the learning outcomes, resulting in stronger learning motivations. **Task 3** attempts to order multiple courses towards getting relevant job opportunities.

1.3.2 Providing a systematic view on course content

Regarding the analysis of course content, a line of research attempted to extract and build concept maps from learning materials [27, 28, 29], or expand the concepts to complement the original course content [30]. To encourage the retrieval and reuse of learning materials, another line of research tried to annotate the learning materials with metadata [31] or external standards [32].

These works focused on the retrieval and organization of detailed concepts, which results in an over-exposure of the content and a lack of systematic view. Suppose that a student is interested in learning “sorting algorithm”, then previous research works well in identifying a sequence of learning materials teaching related concepts such as “selection sort”, “insertion sort”, “quicksort”, and “mergesort”. However, we encourage the student to take a step back and look from a bigger picture. In other words, we want to provide answers to questions such as “What problems are usually associated with sorting algorithms?”, “Are there any other algorithms?”, “How do sorting algorithms relate to data structures and specific programming languages?”, and so on.

Task 1 attempts to solve this problem by utilizing a domain knowledge categorization. Understanding how the courses contain knowledge over different categories enables the students to explore and compare courses, which also contributes to the refinement of their learning goals.

1.3.3 Quantifying knowledge coverage

For specific learning goals such as acquiring the knowledge in predefined categories, a quantitative estimation of knowledge coverage helps select relevant courses and reveal knowledge gaps. Some works attempted to discover whether the knowledge category of interest is covered or not by academic programs or courses [33, 32]. While other works took a further step to inspect the extent to which knowledge categories are covered [34, 35, 36]. To the best of our knowledge, **Task 2** is the first work to provide a rigorous definition of coverage and estimate it in a semi-automatic manner. The estimated coverage can be used as an effective criterion to select appropriate courses.

1.3.4 Modeling a rank-sensitive ordering problem

When recommending learning materials, many aspects need to be considered such as the underlying dependency relationships, the availability, and the importance/relevancy of the materials. Therefore, the core of such problems is to balance the trade-off among different aspects. Some works have attempted to recommend a set of courses with constraints [37, 20], and recommend learning paths of concepts given the start and end ones [38, 39]. Unlike these works, **Task 3** aims at identifying a rank-sensitive ordering of courses while paying attention to the prerequisite relationships. In other words, our work ensures that the information gain at every position of the order is optimized, which is practically helpful for maintaining students' motivations.

1.4 Thesis Structure

The remainder of this thesis is organized as follows: In Chapter 2, we model the course content by matching it to a domain knowledge categorization. Specifically, we propose a Wikipedia structure-based method to capture the implicit relatedness of a course and a domain knowledge category. In Chapter 3, we estimate the knowledge coverage of a knowledge category by the courses based on a taxonomy and the idea of centrality. In Chapter 4, we order the courses toward getting relevant job opportunities while following the prerequisite relationships among the courses. We then conclude the thesis in Chapter 5.

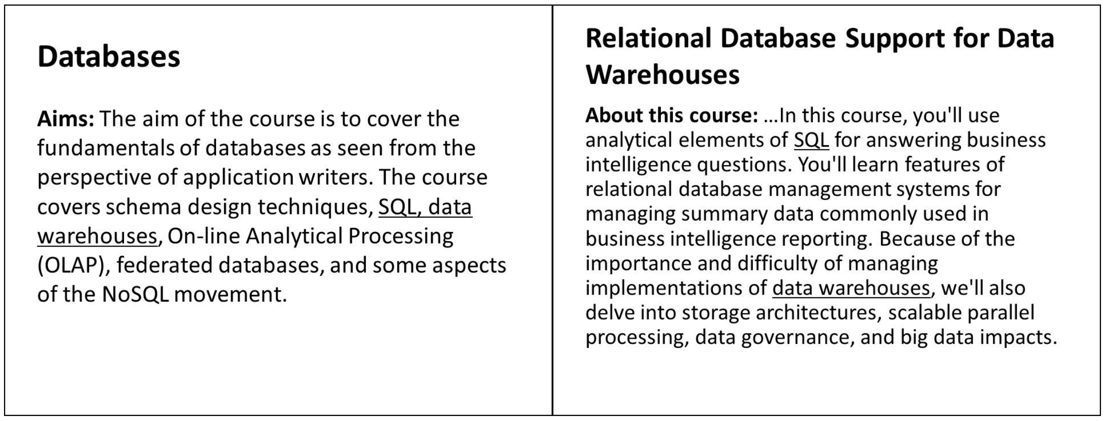
CHAPTER 2

Course Content Modeling

As described in Chapter 1, it is difficult for the students to understand the course content given the existing techniques such as course concepts and course metadata, especially when they do not have a specific learning goal. In this chapter, we approach this problem by leveraging the domain knowledge categorization, which demonstrates the overall structure of the knowledge in the domain. Identifying how courses teach the knowledge over those categories can help refine the unclear learning goals and make a comparison between different courses.

2.1 Introduction

In traditional higher education, the courses that students can receive over their lifetime are significantly constrained by their regional, religious and economic status. Fortunately, the appearance of Massive Open Online Courses (MOOCs) provides opportunities to those who are longing for knowledge. According to Dillahunt et al.'s survey [40] on six MOOCs offered by the University of Michigan from fall 2012 through winter 2013, the majority of their MOOC users possessed at least one higher education degree, which means that MOOCs are actually utilized as a supplement to traditional educational programs. In addition, among the users who completed any one of those MOOCs, the users without access to traditional education demonstrated a higher rate of completion with distinction performances than those with access to traditional edu-



Databases

Aims: The aim of the course is to cover the fundamentals of databases as seen from the perspective of application writers. The course covers schema design techniques, SQL, data warehouses, On-line Analytical Processing (OLAP), federated databases, and some aspects of the NoSQL movement.

Relational Database Support for Data Warehouses

About this course: ...In this course, you'll use analytical elements of SQL for answering business intelligence questions. You'll learn features of relational database management systems for managing summary data commonly used in business intelligence reporting. Because of the importance and difficulty of managing implementations of data warehouses, we'll also delve into storage architectures, scalable parallel processing, data governance, and big data impacts.

(a) Course I (b) Course II

Figure 2.1: Two syllabi of courses related to “database”.

cation. This indicates the potential of empowering the under-educated population with open educational resources such as MOOCs.

The MOOC environment brings changes to both students and educators. For students, elaborately designed curricula are forced upon them in traditional higher education, whereas MOOC environments allow them to be more active in choosing courses for their own learning purposes. For educators, the competition that usually occurs at the level of the institution, program or curriculum in traditional higher education switches to the level of an individual course. Therefore, it will be of paramount importance for both students and educators to identify the contents of massive courses in an agile manner.

At this moment, one of the largest MOOC platforms— Edx provides more than 2,500 courses in cooperation with more than 140 educational institutions worldwide.[2] Since these courses are neither designed for a unified educational purpose nor presented using a controlled vocabulary, it is difficult to recognize the commonalities and differences among these courses. Figure 2.1 shows two syllabi of courses related to the subject “database”. Resemblances such as “SQL” and “data warehousing” can be captured solely from the textual information of two syllabi. However, these shallow resemblances are not sufficient to answer questions that may be raised by students such as “Do I need to take both of them?”, “In which order should I take them?”, and “Are these two courses enough if I want to be a database professional?”. To better understand the implicit

[2]`https://www.edx.org/`, accessed November 25, 2020.

Table 2.1: Instances of *KA*, *KU* and *Topic* in CS2013.

KA	***KU***	***Topic***
Information Management	Database Systems	Approaches to and evolution of database systems
		Components of database systems
		...
	Data Modeling	Data Modeling
		...
	...	...
...	...	...

relationships among courses and the relative positions of courses in the domain, we consider a standard categorization of domain knowledge as being necessary.

"Computer Science Curricula 2013" (CS2013) [41], published by the joint task force of the Association for Computing Machinery (ACM) and IEEE Computer Society, provides an example of a standard categorization of knowledge in the domain of computer science. This curricular guideline has a history of more than 40 years and aims at guiding diverse educational institutions on what knowledge should be covered in an undergraduate program in computer science. In CS2013, the domain knowledge is presented as a set of *Topics*, grouped into 163 *Knowledge Units* (*KUs*). These *KUs* are further grouped into 18 *Knowledge Areas* (*KAs*). As a result, we can observe broad subjects, median topics and detailed concepts of this domain from this *KA-KU-Topic* structure. Table 2.1 shows partial content of *KA:Information Management* and its *KUs*, *Topics*.

If we succeed in identifying which domain knowledge categories and to what extent they are taught in each course, the underlying differences and compatibility between courses will become apparent. Taking the courses in Figure 2.1 for example, suppose that we have successfully inferred how Course I and Course II cover the knowledge in each domain knowledge category. In Figure 2.2, we can observe that Course I briefly but widely covers some basic knowledge in information management such as "Database Systems", "Data Modeling", "Relational Database" and "Query Language". Meanwhile, Course II puts greater emphasis on "Query Language" and "Physical

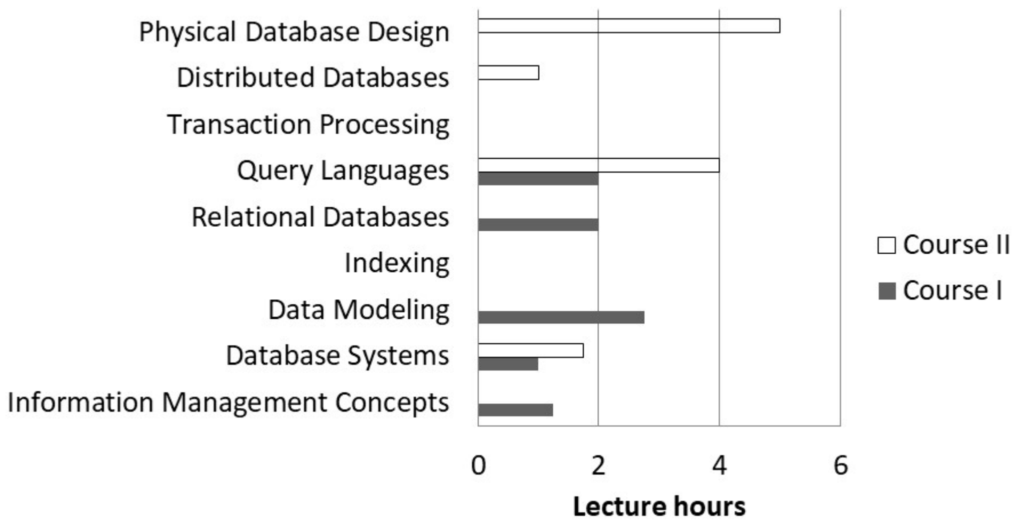


Figure 2.2: Knowledge distribution of course content. The vertical axis refers to the *KUs* in *KA:Information Management* and the horizontal axis represents the lecture hours spent on each *KU*.

Database Design", which demonstrates a special interest in database design. Such knowledge distribution information is instructive when judging which course better accommodates a student's interest. Therefore, we establish the task of extracting course knowledge distribution over the domain knowledge categories defined in CS2013 as the main goal of this work.

In the literature that aims at supporting online learning, some works [42, 43, 44, 27, 28, 29] attempted to generate a concept map as a representation of course content. Since their primary concern is to support course learning or course design, the concept map presents many detailed concepts within a course. This is not suitable for implying course content from a macro perspective of this domain. Other works put emphasis on the reuse and retrieval of learning materials, and they attempted to annotate a piece of learning material with some external standards. One example of such standards is "IEEE Learning Object Metadata Standards" [31]. Although this metadata standard contains several educational items (e.g., interactivity type and intended end user role), it does not dig into the content of the learning materials, which makes it insufficient for the identification of course content. Other works [35, 45, 32, 36] have attempted to match course materials with domain knowledge standards for curriculum or course content analysis. Our problem setting is more difficult than their settings from several aspects, as will be discussed in Section 2.2.2.

To extract the knowledge distribution of a course over the domain knowledge categories, we utilize the course syllabus as the description of the course content and the text in CS2013 as the description of the domain knowledge category *KU*. Then, we address this problem as estimating the relatedness of the syllabus and the description of a domain knowledge category. Intuitively, we employ a bag-of-words method—Labeled Latent Dirichlet Allocation (LLDA) to infer the probabilities that a syllabus is related to the domain knowledge categories. Nonetheless, the lack of contextual information in the bag-of-words method may be intensified by the fact that both the syllabus and the description of the domain knowledge category are written in short texts. To solve this problem, we devise a Wikipedia structure-based method to capture the implicit relatedness of a syllabus and the description of the domain knowledge category by bridging them under the Wikipedia article and category structure. Several previous works [46, 47, 48, 49, 50] exploited Wikipedia article and category structure to estimate the relatedness of two concepts or two documents. However, such methods are not suitable for our problem setting, which drives us to propose an original Wikipedia structure-based method in this work.

In our experiment, we utilize the *Topics* of *KU* in CS2013 as the description of the domain knowledge category, and the syllabi of 131 CS-related courses as the reflection of the course content. The information provided by course instructors consisting of how many lecture hours are spent on each *KU* is used as the ground truth. We evaluate the proposed method at two levels according to task difficulty: 1) whether a syllabus teaches a domain knowledge category and 2) to what degree a syllabus teaches the domain knowledge categories. For the latter, the result shows that we can estimate the *KU* distribution of a course at an accuracy rate of 0.537 in terms of cosine similarity. For the more challenging task, i.e., estimating the *KU* distribution of a course, the Wikipedia structure-based method achieves a better performance than the baseline method.

The contributions of this work are twofold:

- We utilize the domain knowledge categorization defined in a curricular guideline to model the knowledge distribution of a course, which is more readable in pedagogical terms compared with the concept maps. In addition, a standard domain knowledge categorization enables the comparison of heterogeneous courses, which is otherwise difficult to achieve using concept maps extracted from individual courses.

- When matching a syllabus and the descriptions of domain knowledge categories, the leverage of the Wikipedia article and category structure can prevent the loss of contextual information under the bag-of-words method and increase accuracy.

2.2 Related Work

We divide the related works into three categories according to how they are related to our work. In Section 2.2.1, we introduce the works that attempt to generate a concept map from the educational content and explain why their approach is not applicable to our problem setting. In Section 2.2.2, a detailed comparison between the works that share similar research goals and ours is conducted. Finally, we summarize relatedness estimation methods that utilize the Wikipedia category system in Section 2.2.3.

2.2.1 Generating concept maps

A concept map is a visualization tool for organizing and representing knowledge [51]. In some works regarding adaptive learning systems, a concept map was used as a means to organize the knowledge of course materials and to provide cues for navigating the student to the "concept" that should be learned next based on their mastery conditions on previous "concepts" [42, 43, 44]. A visually straightforward concept map is considered to be of enormous help when demonstrated to the students during the process of learning. A stream of research [27, 28, 29] aimed at automatically extracting concepts and the relationships between them from course materials. Because the basic intention of such concept maps is to facilitate course design or course learning within a course, the concept map presents many detailed concepts in a course and highlights the relationships among these concepts. If presented in the process of searching a course, this type of concept map may confuse the student, who has not even begun to learn about this course. Instead, a systematic categorization of domain knowledge is desirable to help students identify the content of a course without a comprehensive understanding of the course content. In addition, the concepts extracted from different courses tend to have different representations of the same knowledge due to the diversity in the word choices of the instructors, which leads to difficulties in comparing the content of heterogeneous courses at one time.

Aside from from facilitating course learning and course design, concept maps can be generated for various purposes, including as an intermediate result of certain learning supporting tasks. Agrawal et al. [52] attempted to automatically group and sequence a given set of concepts for supporting course design. As an intermediate output of their research, a prerequisite relationship embedded concept map was constructed and further used in synthesizing a study sequence. Given that their main concern is to form a final study sequence of concepts, their concept map cannot be utilized as an indicator of course content. Yang et al. [53] held a view that a universal concept graph is instrumental in reasoning the course content overlap and dependency, especially in a MOOC environment where the courses are offered by different universities. They adopted an approach for inferring unobserved course dependencies from existing course dependencies. The fact that only course dependencies within a university are available means that their model cannot infer the course dependencies across different universities. Although they mentioned that the intermediate product of their research—the universal concept graph—is suggestive and useful to both students and instructors, the graph's validity lacked a thorough evaluation.

2.2.2 Annotating learning materials

Learning Object Metadata Standards

A "learning object" is defined as any digital material that can be used for educational purposes [31]. To make learning objects much easier to retrieve, use, evaluate, and manage by students, educators and automated software processes, "IEEE Learning Object Metadata Standards" has been published [31]. In this standard, a systematic data scheme for learning objects is presented. In addition to some basic items, such as identifier, title, language, and keyword, education-specific items are also included. However, rather than factors concerning the content of learning objects, only factors that are related to learning styles are discussed in that standard. For instance, "interactivity type" identifies the learning mode supported by the learning object, and "intended end user role" specifies the principal users for who the learning object is designed. These items will not be able to discriminate between two similar courses at the knowledge level.

Table 2.2: A comparison with related works.

	Sekiya et al. [45]	Contractor et al. [32]	Kawintirannon et al. [36]	Our work
Analyzing at the level of course	×	✓	×	✓
Applied for Higher Education	✓	×	✓	✓
Analyzing at the granularity of *KU*	×	✓	✓	✓
Evaluation				
Using ground truth authored by experts	×	✓	×	✓
Evaluating the proportion of knowledge distribution	×	×	×	✓

Mapping Learning Materials with Curricular Guidelines

To the best of our knowledge, academia has not begun to map learning materials with the domain knowledge, such as curricular guidelines, until recently. On one hand, some works [35, 45, 36] utilized curricular guidelines to observe the overall knowledge distribution of curricula from different educational institutions. On the other hand, Contractor et al. [32] attempted to label educational content with a learning standard (a file that specifies the learning outcomes that a student is expected to acquire after completing a course) that works similarly to a curricular guideline such as CS2013. Because Ishihata et al. [35] addressed this problem in a manual manner, their work will be omitted in the following discussion. To highlight the originality of our work, we compare the works of Sekiya et al. [45], Contractor et al. [32] and Kawintirannon et al. [36] with our work from the aspects listed in Table 2.2. In detail, our work distinguishes itself from the related works as follows:

- The works of Sekiya et al. [45] and Kawintirannon et al. [36] emphasized observations on the whole curriculum, whereas our interest lies in the analysis on individual courses, especially on their positions in this domain and their relationships with each other. Their methods can possibly be applied to analyzing learning materials at the level of the individual course, but the accuracy of the result remains unknown.

- Regarding the data used in analysis, we use course syllabus to infer the knowledge distribution of a course, which raises the task difficulty compared with using more course materials [32, 36].

- With respect to the evaluation method, our ground truth is assigned by the course instructors, which is more reliable. Furthermore, we divide the evaluation difficulty into two stages: 1) whether a knowledge category (i.e., *KU*) is detected by our method and 2) whether the relatedness of the knowledge categories to a course estimated by our method agrees with the true knowledge distribution. With such an evaluation framework, a more robust method can be expected.

- We devise a Wikipedia-structure based method to raise the accuracy of mapping course content to domain knowledge categories, which is an innovative attempt in this problem setting.

2.2.3 Utilizing the Wikipedia category system to estimate relatedness

Wikipedia is an online encyclopedia that can be edited by anyone and is the largest knowledge repository in existence. As a knowledge resource, Wikipedia possesses the following advantages, especially for natural language processing tasks:

- **Wikipedia is balanced with respect to the quantity and quality of information.**

  Compared to general websites, Wikipedia includes more technical and academic information. In addition, compared to ontologies such as WordNet, Wikipedia possesses broader and more up-to-date information [47]. In other words, Wikipedia achieves a balance between the quantity and quality of information as a knowledge resource.

- **Wikipedia assigns unique article pages to different meanings of a polysemic word.**

  In Wikipedia, a unique article page is established for a concept or an entity. Furthermore, different meanings of a polysemic word will have their own corresponding article pages. For example, “NP” is separated into several Wikipedia articles, `NP_(complexity)` and `Noun_phrase`, to name a few, to distinguish between different meanings. This characteristic is helpful when addressing disambiguation [54].

- **Wikipedia article pages are managed in a category system.**

  To help manage Wikipedia articles, a category system is utilized. A unique page is established for a category, which only contains links to related categories and articles but does not include any content for itself. Authors can assign an article to categories according to its content or create a new category if necessary. Figure 2.3 illustrates this Wikipedia article and category structure. When utilizing Wikipedia to estimate the relatedness of concepts (terms), this category system provides considerable indications that two concepts may be somewhat related.

Gabrilovich and Markovitch [46] attempted to represent a term or a document as a vector in the space of Wikipedia articles, which was further used in estimating the relatedness of terms or documents. Similarly, Hu et al. [47] attempted to cluster

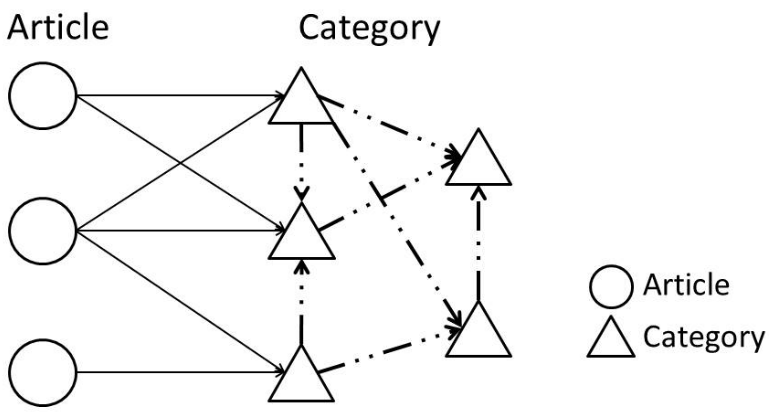


Figure 2.3: Wikipedia article and category structure.

documents by representing them as vectors in the space of Wikipedia categories. In their approach, documents were represented as high-dimensional vectors because a substantial amount of Wikipedia articles (e.g. more than 240,000 Wikipedia articles were used in [46]) and categories were used. As a result, the document vectors tend to be sparse, which produces difficulty in relatedness estimation and increases the computation time.

Genc et al. [48] utilized the Wikipedia article and category structure to estimate distances between tweet messages in a classification task. Despite the fact that our technique is similar to their technique, it is applied to a different end. First, the correctness of the distances between tweets is evaluated with respect to the degree to which they can separate tweets from different classes. Given that they test their method using tweets from three distinctive classes, business news, environment news, and culture news, it is supposed to be relatively simpler than estimating the relatedness of a syllabus and domain knowledge categories. Second, in their distance estimation, only one Wikipedia article is assigned to each tweet. In our case, a syllabus may cover multiple concepts that cannot be represented by only one Wikipedia article. Thus, their method is not suited to our problem.

Strube et al. [49] explored the Wikipedia category structure more intensively. In their research, two terms were first mapped to Wikipedia articles; then, their relatedness was estimated by computing the distances between their associated Wikipedia categories. Treating the Wikipedia category structure as a hierarchy, the semantic distance between two categories is captured based on their positions in that hierarchy. Motivated by a similar task, a generalized flow-based method was proposed by Zhang et al. [50]. Given two Wikipedia articles, they first constructed a network of Wikipedia articles that have page links to these two articles. Then, the strength of the relationship between these two articles was estimated as the maximum information flowing along the paths connecting the articles. The Wikipedia category system was also used to group Wikipedia articles in the network, which helps to discriminate the information gain of the edges of Wikipedia articles from different groups. Despite the fact that these works utilized the Wikipedia category system to estimate the semantic distance between two concepts (i.e., Wikipedia articles), their methods can only be applied to the relatedness estimation of two Wikipedia articles. In our problem setting, a course or a domain knowledge category is supposed to be associated with multiple Wikipedia articles, which makes it unfeasible to adopt their methods.

2.3 Problem Definition

In this work, we address the matching between course content and the domain knowledge categories. Regarding domain knowledge categorization, CS2013 is utilized as a standard categorization of knowledge in the domain of CS. As described in Section 2.1, the knowledge of this domain is represented in a *KA-KU-Topic* structure. Note that the predominant formats of *Topic* are concise sentences or even unstructured sentences (phrases). This derives from the fact that a curriculum guideline essentially functions as a reference of what knowledge is to be included in a curriculum instead of how specifically to deliver the knowledge.

Course content is presented in diverse formats, such as textbooks, slides, videos, and audios. Even for the same subject, different formats can be adopted based on the preferences of instructors. However, regardless of how the instructors present their course content, a syllabus is always created to summarize and introduce the course content to potential course takers. Despite the fact that the information in the syllabus does not represent the course content in full, we view the syllabus as a more subjective reflection of course content and the task of inferring the knowledge distribution of a course solely from the syllabus as a challenge.

Considering the fact that the construction of a syllabus is not always a precise reflection of the actual lecture hour distribution over different knowledge categories (e.g., topics that are addressed over different numbers of lecture hours may be discussed in the same amount of text in a syllabus), we divide our target into two levels according

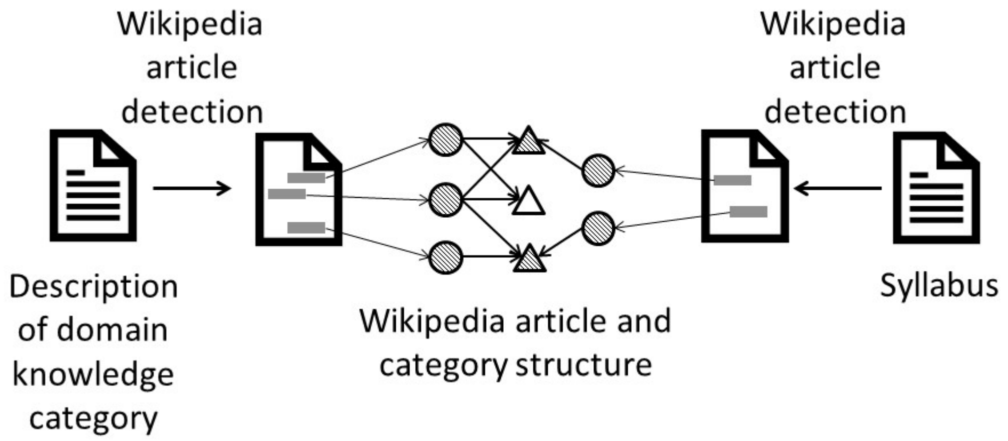

Wikipedia article detection
Description of domain knowledge category
Wikipedia article and category structure
Wikipedia article detection
Syllabus

Figure 2.4: Framework of Wikipedia structure-based method.

to task difficulty:

1. Detecting whether a course covers a specific knowledge category *KU*.
2. Estimating to what extent a course covers a set of knowledge categories *KUs* relatively.

2.4 Methodology

The task of extracting the knowledge distribution of a course can be addressed by estimating the relatedness of a syllabus and the descriptions of domain knowledge categories. As the baseline method, we adopt a mature bag-of-words method— LLDA to infer the probabilities that a syllabus is related to domain knowledge categories. Nonetheless, the defect brought about by the lack of contextual information in the bag-of-words method may be intensified by the fact that both the syllabus and the description of the domain knowledge category are written with a limited number of sentences. Thus, we devise a Wikipedia structure-based method to capture the implicit relatedness of a syllabus and the descriptions of domain knowledge categories by bridging them via the Wikipedia article and category structure. Figure 2.4 shows the framework of the Wikipedia structure-based method.

2.4.1 The baseline method—Labeled Latent Dirichlet Allocation (LLDA)

In the bag-of-words method, a document is captured as a vector in the space of its constituting terms, and each entry indicates the frequency of a term in the given document. LLDA [55] is one of the bag-of-words methods that address topics in documents. In LLDA, the topic of a text is called the "label" of the document. LLDA learns term-label correspondences from a set of labeled documents and infers the probabilities that how likely an unlabeled document is related to those labels. In our setting, the description of the domain knowledge category with its title is viewed as a labeled document to be used in the training set. A syllabus is viewed as a document whose labels await to be inferred.

2.4.2 Wikipedia structure-based method

Intuition

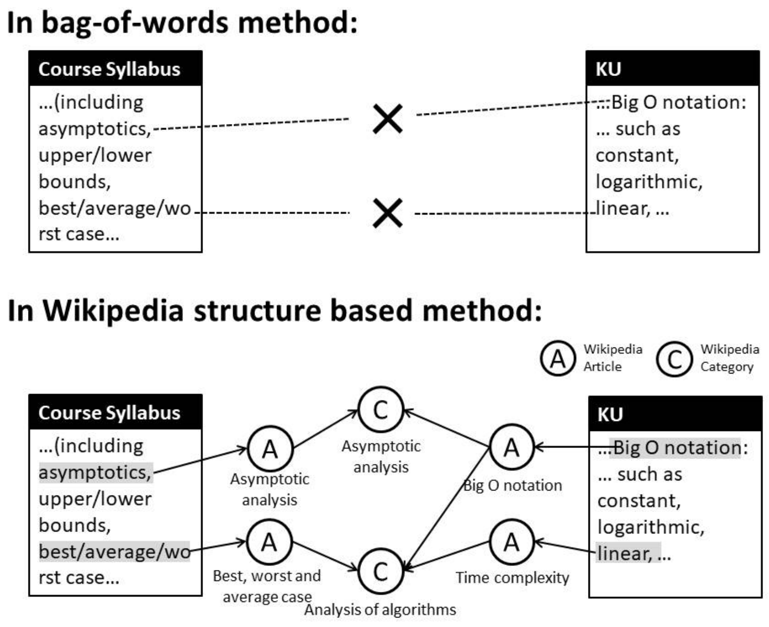
The bag-of-words method emphasizes the frequency of terms in a document while ignoring their contexts. This ignorance of contextual information causes two problems, which can be addressed by leveraging the Wikipedia article and category structure:

- **Term variation.**

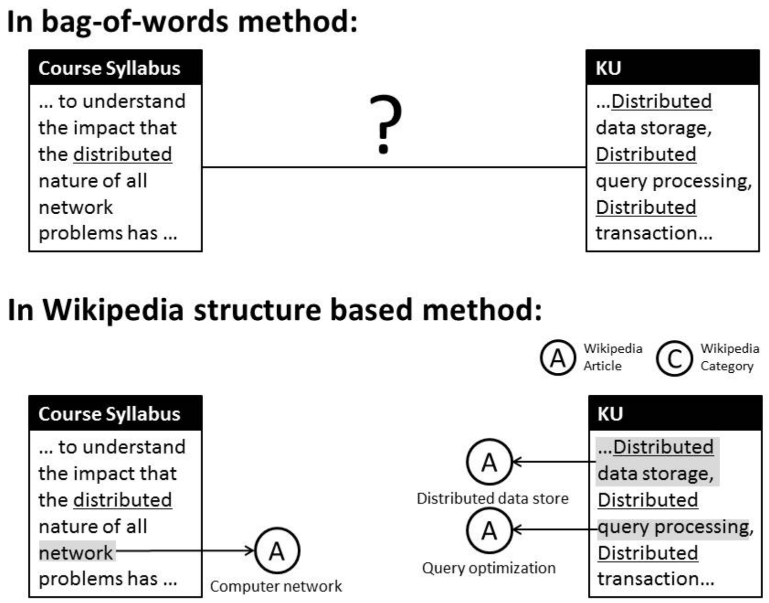
  Related concepts may be presented in completely different terms. Figure 2.5a uses a course related to algorithm and *KU:Algorithm and Complexity/Basic Analysis* as examples. It is desirable for these two documents to be recognized as related. However, as shown in the upper part of Figure 2.5a, they will be considered as unrelated since they share no common terms. In comparison, as shown in the lower part of Figure 2.5a, if we bridge the two documents by extracting their associated Wikipedia articles and Wikipedia categories, their implicit relatedness can be captured and quantified by computing their distance in the Wikipedia-structure (the detailed computation will be discussed in the next section).

- **Term ambiguity.**

  A term may have different meanings and emphasis in different contexts. Figure 2.5b uses a course related to computer network and *KU:Information Management/Distributed Databases* as examples. Clearly, this course is not supposed to


In bag-of-words method:
Course Syllabus
...(including asymptotics, upper/lower bounds, best/average/worst case...
KU
...Big O notation: ... such as constant, logarithmic, linear, ...
In Wikipedia structure based method:
Wikipedia Article
Wikipedia Category
Asymptotic analysis
Asymptotic analysis
Big O notation
Best, worst and average case
Analysis of algorithms
Time complexity

(a) Address term variation.


In bag-of-words method:
Course Syllabus
... to understand the impact that the distributed nature of all network problems has ...
?
KU
...Distributed data storage, Distributed query processing, Distributed transaction...
In Wikipedia structure based method:
Wikipedia Article
Wikipedia Category
Computer network
Distributed data store
Query optimization

(b) Address term ambiguity.

Figure 2.5: Intuition of Wikipedia structure-based method.

be related to a database *KU*. However, as shown in the upper part of Figure 2.5b, they are prone to being recognized as related under the bag-of-words method simply because they include a common term—"distributed". The associated Wikipedia article detection helps to reduce the effect of a single term. As illustrated in the lower part of Figure 2.5b, "distributed" alone does not have a corresponding Wikipedia article, whereas "distributed data storage" is related to a database-related Wikipedia article (the detailed Wikipedia article detection will be discussed in the next section). As a result, these two documents possess different associated Wikipedia articles, which results in being correctly recognized as unrelated.

Procedure

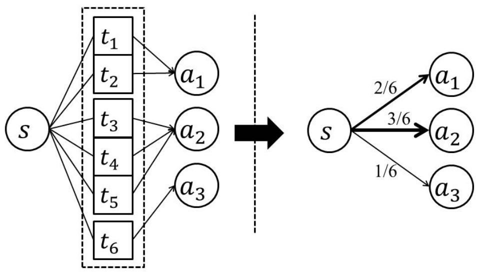
To estimate the relatedness of two documents based on the Wikipedia article and category structure, we perform the following steps.

Step 1: Detect associated Wikipedia articles. For a given document, we use two methods to extract the Wikipedia articles that are associated with the content of this document.

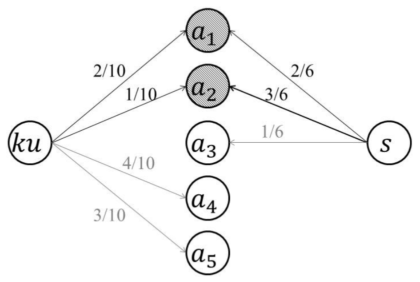
1. Use DBpedia Spotlight [56], a tool that automatically detects the terms or phrases that have a corresponding Wikipedia article and returns the article URLs for a given text. For each term or phrase, only one associated Wikipedia article will be selected. Based on our empirical observation, this tool achieves a relatively high precision and low recall in retrieving target Wikipedia articles.
2. Query the noun phrases of the document in a search engine with the URL limited to "Wikipedia.org". We utilize the Natural Language Toolkit[3] to extract noun phrases in the document and search the associated Wikipedia articles for them using the Bing search API[4]. Similar to DBpedia Spotlight, we only choose the first Wikipedia article returned by the search engine for each phrase. For the noun phrases in the description of the domain knowledge category, we add the title of *KU* to it to complement some contextual information before submitting it to the search engine.

[3]http://www.nltk.org/

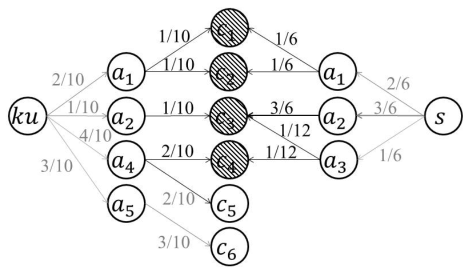
[4]https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/



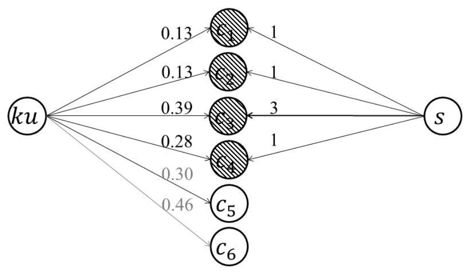
(a) Computing $con(a,s)$. E.g., $T^s_{a_1} = \{t_1, t_2\}$, $con(a_1,s) = 2/6$.



(b) Computing $rel_1(s,ku)$. E.g., $rel_1(s,ku) = con(a_1,s)+con(a_2,s)+con(a_1,ku)+con(a_2,ku) = 2/6+3/6+2/10+1/10$.



(c) Computing $rel_2(s,ku)$. E.g., $rel_2(s,ku) = con(c_1,a_1^{ku}) + con(c_1,a_1^s) + \ldots + con(c_4,a_4^{ku}) + con(c_4,a_3^s) = 1/10+1/6+\ldots+2/10+1/12$.



(d) Computing $rel_3(s,ku)$. E.g., $rel_3(s,ku) = con(c_1,ku) \times con(c_1,s) + \ldots + con(c_4,ku) \times con(c_4,s) = 0.13 \times 1 + \ldots + 0.28 \times 1$.

Figure 2.6: Computing the relatedness of two documents only using the Wikipedia article and category structure.

Note that a discussion of the information retrieval techniques embedded in the above two methods is beyond the scope of this paper. Briefly, these two methods are based on a framework that retrieves a Wikipedia article if its content shares information with the query.

Step 2: Compute the relatedness of two documents. As illustrated in Figure 2.3, a Wikipedia category can be assigned as a parent or child category of another Wikipedia category without constraints, which leads to loops and a complex network of Wikipedia categories. To reduce noise, we only consider the level of the Wikipedia article, the first level of the Wikipedia category, and the article-category relationship between these two levels. Then, we apply three patterns to compute the relatedness of two documents. Let s be a course syllabus, ku be a

KU, and $rel_i(s, ku)$ be the relatedness of s and ku computed in Pattern i.

(Pattern 1) Only consider the level of the Wikipedia article.

We suppose that if two documents have many associated Wikipedia articles in common, they are highly related. Then we compute $rel_1(s, ku)$ as

$$rel_1(s, ku) = \alpha \times (\sum_{a \in A_s \cap A_{ku}} con(a, s) + con(a, ku)) \,, \tag{2.1}$$

where α is an amplifier and will be discussed in Step 3. A_s and A_{ku} denote the set of Wikipedia articles that are associated with s and ku, respectively. $con(a, \cdot)$ denotes the contribution of a Wikipedia article a to the document. We then compute $con(a, s)$ and $con(a, ku)$ as

$$con(a, s) = \frac{|T_a^s|}{\sum_{a \in A_s} |T_a^s|} \,, \tag{2.2a}$$

$$con(a, ku) = \frac{|T_a^{ku}|}{\sum_{a \in A_{ku}} |T_a^{ku}|} \,, \tag{2.2b}$$

where T_a^s and T_a^{ku} denote the set of terms that have the same associated Wikipedia article in s and ku, respectively. Figure 2.6a illustrates this computation.

In Equation (2.1), we choose to sum $con(a, s)$ and $con(a, ku)$ rather than to multiply them for two reasons: 1) Because we conduct completely identical computations on the contribution of a Wikipedia article toward a syllabus and *KU*, it is reasonable to sum up the $con(a, s)$ and $con(a, ku)$ of a common Wikipedia article. 2) By multiplying $con(a, s)$ and $con(a, ku)$, it is highly possible that the contributions of $con(a, s)$ or $con(a, ku)$ toward $rel_1(s, ku)$ will be emphasized if the number of associated Wikipedia articles of the syllabus or *KU* varies across different syllabi or *KUs*, which is not desirable in our problem setting. We confirm in the preliminary experiment that summing $con(a, s)$ and $con(a, ku)$ achieves better results than does multiplying them. Figure 2.6b illustrates the computation of $rel_1(s, ku)$.

(Pattern 2) Consider the level of the Wikipedia article and the first level of the Wikipedia category.

Except for the associated Wikipedia articles in common, if two documents have shared Wikipedia categories via different associated Wikipedia articles, we suppose that they are also possibly related. As the example shows in Figure 2.5a, the syllabus and the description of the domain knowledge category do not share any common Wikipedia articles, but they are related if we involve the level of the Wikipedia category. However, compared with the associated Wikipedia articles, the shared Wikipedia categories indicate an indirect relationship between two documents, whose contributions should be discounted in the computations. Let a^s be a Wikipedia article in A_s, and C_{a^s} be the set of Wikipedia categories connected to a^s. Similarly, Let a^{ku} be a Wikipedia article in A_{ku}, and $C_{a^{ku}}$ be the set of Wikipedia categories connected to a^{ku}. $con(c, a^s)$ and $con(c, a^{ku})$ denote the contribution of a Wikipedia category c toward a Wikipedia article a that is associated with s and ku, respectively. We then compute $con(c, a^s)$ and $con(c, a^{ku})$ as

$$con(c, a^s) = \frac{con(a^s, s)}{|C_{a^s}|}, \tag{2.3a}$$

$$con(c, a^{ku}) = \frac{con(a^{ku}, ku)}{|C_{a^{ku}}|}. \tag{2.3b}$$

Then, we compute $rel_2(s, ku)$ in Equation (2.4). Note that the sum of the $con(c, a^s)$ and $con(c, a^{ku})$ of a in $A_s \cap A_{ku}$ equals Equation (2.1). The difference with Pattern 1 lies in the integration of $con(c, a^s)$ or $con(c, a^{ku})$ whose a is not in $A_s \cap A_{ku}$.

$$\begin{aligned} rel_2(s, ku) &= \alpha \times (\sum_{\substack{c \in C_s \cap C_{ku} \\ a^s \in A_s \\ a^{ku} \in A_{ku}}} con(c, a^s) + con(c, a^{ku})) \\ C_s &= \bigcup_{a^s \in A_s} C_{a^s} \\ C_{ku} &= \bigcup_{a^{ku} \in A_{ku}} C_{a^{ku}} \end{aligned} \tag{2.4}$$

We choose to sum $con(c, a^s)$ and $con(c, a^{ku})$ rather than to multiply them for two reasons: 1) Because we conduct completely identical computations on $con(c, a^s)$ and $con(c, a^{ku})$, it is reasonable to add $con(c, a^s)$ and $con(c, a^{ku})$ for a common Wikipedia category. 2) Because one Wikipedia category may be connected to different numbers of Wikipedia articles from A_s and A_{ku}, the asymmetry of connections between Wikipedia categories and Wikipedia articles makes it unreasonable to multiply $con(c, a^s)$ and $con(c, a^{ku})$. Figure 2.6c illustrates this computation.

(Pattern 3) Only consider the first level of the Wikipedia category.

Let $con(c, ku)$ be the contribution of a Wikipedia category c toward ku, which should be different even via the same Wikipedia article. Therefore, we adopt the TF-IDF scheme [57] to discriminate $con(c, ku)$, which means that a Wikipedia category will be considered important for a document only if it is associated with this document frequently while also being seldom associated with other documents. First, we generate a KU-C matrix whose entries represent the frequency at which a Wikipedia category is associated with a *KU*. Then, we apply the TF-IDF scheme to transform the original entries into TF-IDF values that will be regarded as $con(c, ku)$. For syllabus s, we do not conduct this process because in a real scenario, our method should be able to estimate knowledge distribution for a syllabus individually. Thus, we simply use the frequency of a Wikipedia category c that is associated with it as $con(c, s)$. Finally, $rel_3(s, ku)$ is computed as

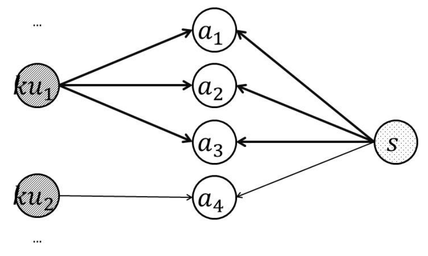
$$rel_3(s, ku) = \alpha \times (\sum_{c \in C_s \cap C_{ku}} con(c, s) \times con(c, ku)) . \tag{2.5}$$

Here, we choose to multiply $con(c, s)$ and $con(c, ku)$ because they are computed in different processes and are not in the same scale, which makes them unsuitable to being summed up. In the preliminary experiment, we confirm that multiplying these two elements works better than summing them with a normalization. Figure 2.6d illustrates this computation.

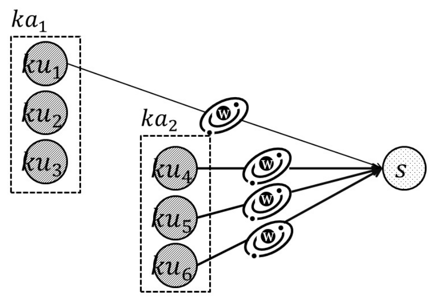
Step 3: Based on the following two elements, add three types of amplifiers to the contributions of the Wikipedia articles and Wikipedia categories toward the documents.

(Element a) We suppose that if two documents share a greater number of associated Wikipedia articles, they are related at a higher probability. As shown in Figure 2.7a, the relatedness of s with ku_1 is more reliable than that to ku_2 because the former is supported by a greater number of associated Wikipedia articles. We call this factor Element a and quantify it as $element_a = |A_s \cap A_{ku}|$.

(Element b) The *KA*-*KU* structure defined in CS2013 indicates that the *KUs* belonging to the same parent *KA* are intentionally grouped together based on educational purpose, which supports the assumption that a course may inevitably cover



(a) Element a



(b) Element b

Figure 2.7: Two elements of the amplifier.

KUs belonging to a *KA* other than the *KUs* distributed over different *KAs*. As shown in Figure 2.7b, the relatedness of s to (ku_4, ku_5, ku_6) belonging to ka_2 is more reliable than that to ku_1 because the former is supported by more connections with sibling *KUs*. We call this factor Element b and quantify it as $element_b = \sum_{ku \in KU_{ka}} \mathbb{1}\{|A_s \cap A_{ku}| > 1\}$, where KU_{ka} denotes the set of *KUs* that belong to the same parent *KA* ka. Here, $\mathbb{1}\{|A_s \cap A_{ku}| > 1\}$ is an indicator function, which computes whether s and ku have more than one common Wikipedia articles.

To emphasize the functions of Element a and Element b in the computation of the relatedness of two documents, we treat them as the exponents in functions. As a result, the relatedness of a syllabus and a *KU* will be amplified if the syllabus shares many associated Wikipedia articles with this *KU*. Similarly, the relatedness of a syllabus and a *KU* will be amplified if the syllabus is also related to other *KUs* that belong to the same parent *KA*. We form three types of amplifiers and induce them to the coefficient α as follows:

No amplifier: $\alpha = 1$

Amplifier 1: $\alpha = 2^{element_a - 2}$

Amplifier 2: $\alpha = 2^{element_b - 2}$

Amplifier 3: $\alpha = 2^{element_a + element_b - 4}$

2.5 Experiment

2.5.1 Dataset

We utilize the text of *Topics* under a *KU* as the description of this *KU*. Regarding the description of course content, we collect the syllabi of 131 courses whose lecture hours over each *KU* are annotated by their instructors. More specifically, 115 of these courses are from CS2013 and 16 courses are from the bachelor program of computer science in Thompson Rivers University[5]. At last, We have the descriptions of 163 *KUs* and the syllabi of 131 CS-related courses with their lecture hour distributions over *KUs*.

2.5.2 The baseline method

As mentioned in Section 2.4.1, we adopt the LLDA model as the baseline method. In LLDA, the training set used to model the term-label correspondences has a significant influence on the inferred results. As can be observed in Table 2.1, the domain knowledge presented in CS2013 is formatted in concise or even incomplete sentences, which leads to a lack of sufficient discriminating terms for a label. Although we can add external information to the original text of a label, the added terms cannot guarantee a higher capability of the model to correctly infer the probabilities that a new document is related to the label. The reason lies in the fact that the bag-of-words method treats each term independently and equally. This loss of contextual information produces difficulties in addressing term variations and term ambiguity. To verify the superiority of our proposed method, we also resort to Wikipedia to extend the description of the domain knowledge category for the LLDA method. Specifically, for each *KU*, we utilize DBpedia Spotlight [56] to extract associated Wikipedia articles and then append their abstracts to the original description of this *KU*. Therefore, we are able to compare the effectiveness of two methods which both leverage Wikipedia.

In the Wikipedia structure-based method, we use Element b to amplify the relatedness of a syllabus with a *KU* when the syllabus is also related to many *KUs* belonging to the same parent *KA*. In LLDA, we can achieve this effect by manipulating the documents in the training set as follows:

1. **1-phase manner**

[5]`https://www.tru.ca/science/programs/compsci/programs/cs_bachelor_of_computing_science.html`, accessed January 18, 2021.

Input the descriptions of all the *KUs* into the training set and infer the probabilities that a syllabus is related to these *KUs* directly.

2. **2-phase manner**

   In the first phase, we simply concatenate the descriptions of *KUs* of one *KA* as its description. Then, we estimate the probabilities that a syllabus is related to the *KA* and choose the top k *KAs* with the highest probabilities. In the second phase, we estimate the probabilities that a syllabus is related to the *KUs* which belong to the chosen *KAs*. k is empirically set to 3, which is the average number of *KAs* that the courses are associated with.

2.5.3 Evaluation metric

We adopt two evaluation metrics according to the difficulty levels of our task:

1. **Area Under the Curve (AUC)** for evaluating whether a course is correctly detected to cover a knowledge category.

   AUC is a commonly used metric in statistics to evaluate a classifier. It is computed as the area under a Receiver Operating Characteristics curve (ROC) that plots the true positive rate against the false positive rate at various threshold values. In a word, AUC evaluates the ability of a classifier to rank positive instances higher than negative instances.

2. **Cosine Similarity** for evaluating to what degree the lecture hour distribution of a course over domain knowledge categories is correctly estimated.

   Let v_{pred} and v_{gt} be the estimated vector of relatedness and the annotated lecture hours over the *KUs*, respectively. Then we evaluate the performance by computing $cos\_sim(v_{pred}, v_{gt}) = \frac{v_{pred} \times v_{gt}}{||v_{pred}|| \times ||v_{gt}||}$.

2.5.4 Results

For the baseline method, there are two variations, whether one uses the abstracts of the Wikipedia articles to extend CS2013 and whether one conducts the training process in a 1-phase manner or 2-phase manner, which leads to four combinations of experiments.

For the Wikipedia structure-based method, there are three variations, two methods to detect associated Wikipedia articles for a syllabus or the description of the domain

Table 2.3: Experimental results.

		AUC	cos_sim
The baseline method (LLDA)			
1-phase	CS2013	0.746	0.363
	CS2013_wikiabs	0.779	0.414
2-phase	CS2013	0.803	0.383
	CS2013_wikiabs	0.821	0.444
Wikipedia structure-based method			
Syllabus (DBpedia Spotlight)- CS2013 (DBpedia Spotlight)			
No amplifier	Pattern 1	0.744	0.319
	Pattern 2	**0.834**	**0.453**
	Pattern 3	**0.833**	0.429
Amplifier 1	Pattern 1	0.756	0.432
	Pattern 2	**0.832**	**0.481**
	Pattern 3	**0.836**	**0.463**
Amplifier 2	Pattern 1	0.760	0.396
	Pattern 2	**0.844**	**0.510***
	Pattern 3	**0.844**	**0.478**
Amplifier 3	Pattern 1	0.763	**0.459**
	Pattern 2	**0.843**	**0.484**
	Pattern 3	**0.845**	**0.469**
Syllabus (DBpedia Spotlight)- CS2013 (NP+ Bing search)			
No amplifier	Pattern 1	0.709	0.383
	Pattern 2	**0.828**	**0.483**
	Pattern 3	**0.825**	**0.469**
Amplifier 1	Pattern 1	0.712	0.418
	Pattern 2	**0.829**	**0.483**
	Pattern 3	**0.829**	**0.480**
Amplifier 2	Pattern 1	0.714	0.420
	Pattern 2	**0.837**	**0.537***
	Pattern 3	**0.835**	**0.514***
Amplifier 3	Pattern 1	0.714	0.436
	Pattern 2	**0.837**	**0.484**
	Pattern 3	**0.836**	**0.484**

The cells colored are selected as the best baseline results.
The values in bold are better results than the best baseline.
* The values are significantly ($p \leq 0.05$) greater than the best baseline.

knowledge category, three patterns to compute the relatedness of two documents, and four types of amplifiers, which generates 48 combinations of experiments in total. Due to space limitations, we omit all the experiments of two combinations derived from the first variation, namely, “Syllabus (NP+Bing search)-CS2013 (DBpedia Spotlight)” and “Syllabus (NP+Bing search)-CS2013 (NP+Bing search)”, as their performance is inferior to the other two combinations. Finally, we show 4 experimental results for the baseline method and 24 experimental results for the Wikipedia structure-based method in Table 2.3.

As shown in Table 2.3, the best baseline method is the one leveraging Wikipedia abstracts and estimating probabilities in 2-phase manner, from the perspectives of both metrics. In the Wikipedia structure-based method, most of the experiments in Patterns 2 and 3 show higher values than the best baseline method. Furthermore, for the stricter evaluation metric cos_sim which evaluates the detailed knowledge distribution, the experiments in Patterns 2 and 3 with Amplifier 2 achieve significant higher performance than the best baseline. This indicates that the Wikipedia structure-based method works better in estimating the knowledge distribution over the domain knowledge categories from a limited information source, namely, the syllabus.

Regarding the Involvement of Wikipedia

In the baseline method, we extend the original description of a domain knowledge category with the abstract of the associated Wikipedia article. As can be observed in Table 2.3, the performance of the baseline method using the extended CS2013 increased substantially compared to the performance using the original CS2013, especially for estimating knowledge distribution at the level of the *KU*. We extract the top 20 distinctive terms of *KU:Algorithms and Complexity/Basic Analysis* for the original description and the extended description, respectively. As shown in Table 2.4, the Wikipedia articles provide many effective terms (underlined) that are not contained in the original description in CS2013, which explains why an extension with external information increases the accuracy of estimating knowledge distribution over the domain knowledge categories from course syllabi.

However, even the baseline method using the extended CS2013 is not as competitive as the Wikipedia structure-based method. This may be for the following reasons: 1) Despite the fact that the extension of CS2013 brings many discriminative terms for the domain knowledge categories, the terms may be too detailed to appear in a course syllabus. 2) In the bag-of-words method, only completely identical terms will be captured as related. The extension of CS2013 provides several synonyms for a term, but it cannot provide an exhaustive list of the synonyms that may potentially appear

Table 2.4: The top terms for *KU:Algorithms and Complexity/Basic Analysis* without and with an extension.

CS2013	CS2013_wikiabs
big	notation
notation	function
expected	asymptotic
complexity	complexity
algorithms	omega
little	big
theta	letter
asymptotic	algorithm
logarithmic	time
bounds	growth
constant	algorithms
exponential	recurrence
best	xpx
omega	turing
among	fn
upper	size
master	problems
analysis	alphabet
worst	polynomial
quadratic	equation

in a syllabus. 3) For a polysemic term, the Wikipedia structure-based method detects associated Wikipedia articles based on its surrounding terms, which helps to specify one meaning of the term in this context.

Regarding the Wikipedia Structure-Based Method

For the experiments of the Wikipedia structure-based method, we discuss the impacts of the following variations:

- **Relatedness computation.**

  For Step 2, Patterns 2 (Article and category) and 3 (Category alone) achieve higher performance than Pattern 1 (Article alone). This indicates that the engagement of a deeper Wikipedia-structure can be used to effectively estimate the implicit relatedness between two documents. Moreover, the highest performance is observed in Pattern 2. We consider combining the TF-IDF weighting scheme in Pattern 3 into Pattern 2 as a direction of future work.

- **Amplifier addition.**

  Experiments with Amplifier 2 demonstrate higher performance than other types of amplifiers, as Element b strengthens the relatedness of a syllabus and a *KU* when the syllabus is also related to *KUs* belonging to the same parent *KA*. This indicates that leveraging the *KA*-*KU* structure of the knowledge categories can effectively exclude the noises from accidentally related *KUs*. Amplifier 3, which combines Elements a and b, does not work better than the single-element amplifiers, which implies an overestimation of the amplifier.

2.6 Error Analysis

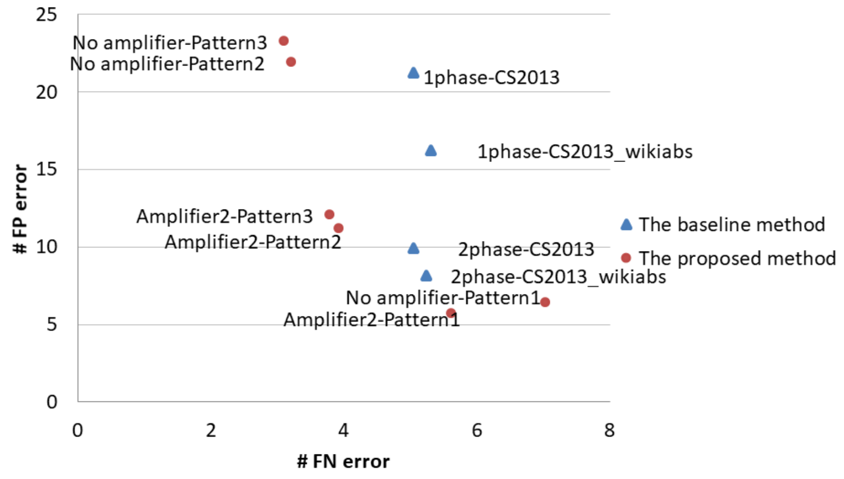
To further identify which factors induced the above results, we conduct both quantitative and qualitative analyses on whether a knowledge category is correctly detected from the syllabus by our proposed method. We utilize the threshold value that leads to the highest ROC value to transform the relatedness of a syllabus and a domain knowledge category into Boolean "0-1" variables. Specifically, we define two types of errors as follows:

False negative error (FN error): For a syllabus, a *KU* that is assigned by the ground truth author but is not detected by our method.

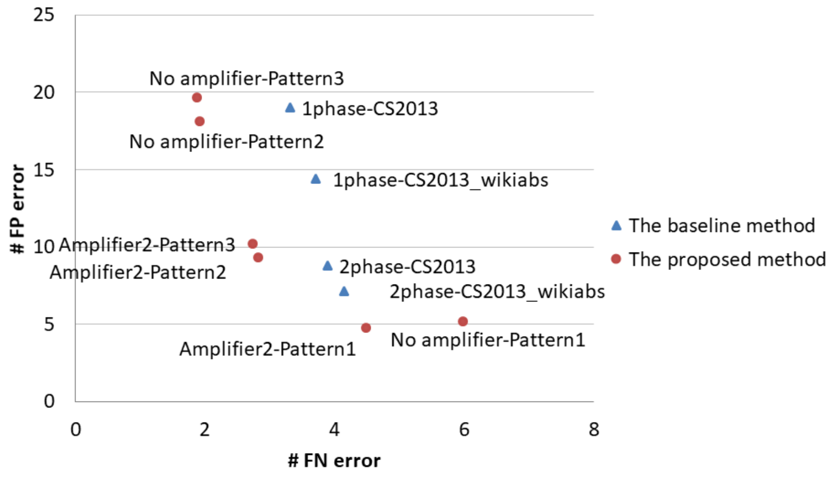
False positive error (FP error): For a syllabus, a *KU* that is not assigned by the ground truth author but is detected by our method.

2.6.1 Quantitative analysis of errors

We choose the experiments framed in Table 2.3 and plot their average numbers of FN and FP errors in one figure. We suppose that FN errors are more fatal than FP errors, since when searching for a course, being unable to find a course containing the target knowledge is more disappointing than finding courses not containing it. To this end, the proposed method in Patterns 2 and 3 works well in reducing FN errors. This is mainly due to the integration of Wikipedia categories which help capture the implicit relatedness. In addition, Amplifier 2 is helpful in reducing FP errors, resulting in a balance between FN and FP errors.



(a) Number of errors.



(b) Number of intolerable errors.

Figure 2.8: Numbers of FN and FP errors generated by the methods.

2.6.2 Qualitative analysis of errors

When investigating on the FN and FP errors by cases, we find that some of the FP errors are undeniable because the “wrong” *KUs* found by our method are closely related to the “correct” *KUs* assigned by the ground truth author. Moreover, when categorizing the knowledge of a domain, overlaps between different knowledge categories are inevitable because they form a common basis for these knowledge categories. It is confirmed that the *KA-KU* structure in CS2013 also contains some overlaps. For example, *KU:Algorithm and Complexity/Algorithmic Strategies* is co-related with *KU:Software*

Development Foundations/Algorithms and Design. There are 121 pairs of such co-related *KUs* across different *KAs* stated in CS2013. We suppose that there should be more such co-related *KUs* because CS2013 is edited collaboratively and it is difficult for a *KU* author to recognize all the other co-related *KUs* in *KAs* edited by other authors. We call the errors induced by co-relationships between *KUs* as tolerable errors, and the remaining errors are considered as intolerable errors. Figure 2.8b shows the number of intolerable errors generated by the methods. We observe a similar trend with Figure 2.8a but less FN and FP errors, which suggests that considering the relatedness among *KUs* may improve the performance.

2.7 Summary

Our work is built upon the premise that mapping course content with a domain knowledge categorization is instrumental in supporting the identification of course content in the process of searching courses. To match course content with domain knowledge categories, we proposed a Wikipedia structure-based method to estimate the relatedness of a syllabus and domain knowledge category by leveraging the Wikipedia article and category structure. Compared with the baseline method, LLDA, we found that our proposed method can effectively capture the implicit relatedness of two short texts. In our experiment, we utilize the *Topic* of *KA*/*KU* in CS2013 as the descriptions of the domain knowledge category and the syllabi of 131 CS-related courses as the representation of the course content. The result shows that we can estimate the *KU* distribution of a course at accuracy rates of 0.837 and 0.537 in terms of AUC and cosine similarity, respectively.

According the the error analyses, we found that our proposed method leveraging Wikipedia categories and *KA*-*KU* structure is balanced at generating FN and FP errors. Improvement on these factors can be a direction of future work. Besides, we found that there exist overlaps among the domain knowledge categories, which induces "tolerable" errors. Consideration on the similarities among knowledge categories is also a future direction.

CHAPTER 3

Knowledge Coverage Estimation

In Chapter 2, we mainly utilize the domain knowledge categories to model the course content, providing a systematic view for the students. In this chapter, we assume a different scenario, in which the students have targeted on a specific knowledge category. In this case, knowing how much of the knowledge is covered in the courses can be a helpful criterion to select courses. We propose a centrality-based method to differentiate the concept importance in the category, which is further used to compute the coverage.[6]

3.1 Introduction

The movement of providing Massive Open Online Courses (MOOCs) emerges from distance education and bursts into popularity in 2012. As is visioned, everyone should be able to access to the course materials on any subject, anywhere, and anytime. Albeit with the difficulty to realize this ideal condition, the current MOOC platforms have brought unprecedented mutual freedom to educators and students. Belanger and Thornton [2] reported that their first MOOC reached around 12,000 students, more than half of whom actually interacted with the course materials. Although only 313 students completed the course successfully, it is noteworthy that those students represented at

[6]This chapter is based on "Estimating Knowledge Category Coverage by Courses Based on Centrality in Taxonomy" [58], which appeared in IEICE Transactions on Information and Systems, Copyright ©2020 IEICE.

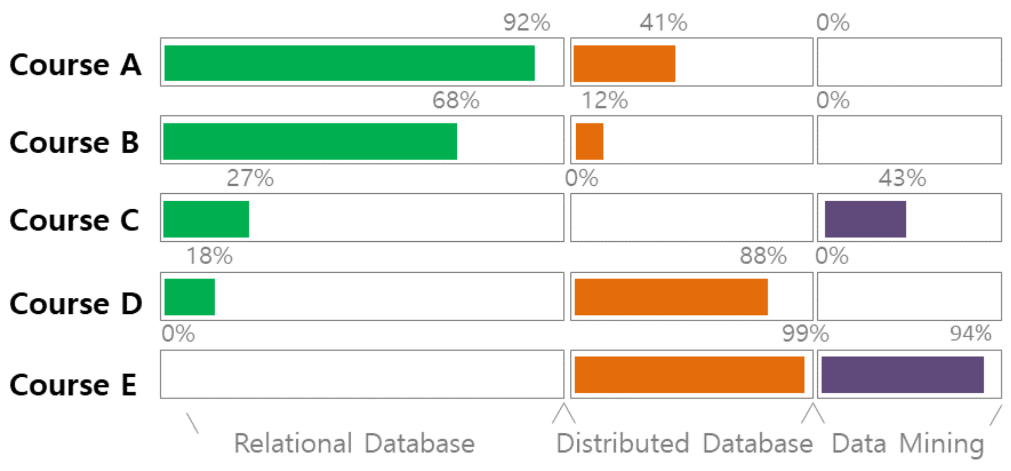


Figure 3.1: Knowledge category coverage and course knowledge composition. The hollow bar indicates the total amount of required knowledge by the category. The colored bar and the percentage value represent how much of the knowledge in a category is covered by a course.

least 37 different countries. It is hardly ever for an instructor in a brick-and-mortar university to reach students with such diverse backgrounds. Meanwhile, the students are faced with various choices of courses offered by different institutions. For example, we have 56 choices on the subject of database in just one of the current MOOC platforms,[7] which are designed and oriented under diverse educational purposes. As a result, it is undoubtedly a difficult task to select the proper course that satisfies one's learning need.

Categorization is an effective way to manage information. Taking "Database" for example, it is such a broad subject that we normally break it down into topics like "Relational Database", "Distributed Database", and "Data Mining", to name a few. With these topics, we can tackle the subject by focusing on one aspect of it at one time. In this study, we term the topics as knowledge categories. We presume the user of MOOC has already targeted on some knowledge categories, then it would be helpful if he/she knows how much the knowledge in the categories is covered by the courses.

[7]`https://www.edx.org/course?search_query=database`, accessed June 19, 2019.

For example, suppose the user is interested in learning "Relational Database". As shown in Figure 3.1, we can rank the courses based on the degree to which they cover the knowledge of "Relational Database". It is straightforward that Course A serves the user's need best since it covers the knowledge of this category with a highest percentage 92%. Additionally, if we are given the absolute amount of knowledge that is required in each category (i.e., the length of each hollow bar in Figure 3.1), we can compare the course knowledge compositions as well. As shown in Figure 3.1, we obtain an overall impression that Course A and B put an emphasis on "Relational Database" and touch some knowledge of "Distributed Database". In contrast, Course E teaches intensively the knowledge of "Distributed Database" and "Data Mining". In this study, we take the first step— estimating the knowledge category coverage as our goal.

Analyzing the course content has been a research interest of education-related communities. Researchers [59, 60, 45] have attempted to understand how the course content distributes over predefined knowledge categories.[8] As mentioned before, our goal in this study— estimating knowledge category coverage— can be extended to acquire course knowledge composition. Other researchers [33, 34, 32, 35, 36] have endeavoured to gauge whether or to what extent a knowledge category is covered by course materials. However, either they employ a manual method or they don't define a concept of coverage. To the best of our knowledge, our study is the first to give a definition of knowledge category coverage and propose a semi-automatic[9] method to estimate it.

To estimate the knowledge coverage of a category by a course, we first model the knowledge category and the course as sets of concepts. Then, we define the coverage as the degree to which the concepts required in a knowledge category are also taught in a course. The key of estimating the coverage is to quantify the importance of concepts to the set, since the importance of the concepts is influenced by the existence of other concepts in the set. We resort to a taxonomy to capture the relationships among concepts and then utilize the idea of centrality to estimate how important a concept is to a set. When applying centrality to our method, we make a special effort to assign larger values to more important concepts without undervaluing less important concepts.

Compared with treating all the concept uniformly important, our centrality-based computation method produces closer coverage values to the ground truth assigned by

[8]Though it may be called as "topic", "knowledge", "academic learning standard", or "knowledge area" in previous research, we unify them into the term "knowledge category" for consistency.

[9]We treat this method as semi-automatic for the reason that one step of the method— the construction of taxonomy is conducted in a manual process.

domain experts. The main contributions of this study are two-fold: 1) Our study is the first one to define a concept of knowledge category coverage and to estimate it in a semi-automatic manner. 2) We construct a taxonomy and utilize the idea of centrality to differentiate the importance of concepts in a set. Moreover, our method is elaborated to weight more important concepts without underestimating other concepts.

3.2 Related Work

3.2.1 Course content analysis

The community of education has a long-standing interest in understanding how knowledge is organized and conveyed in academic programs and courses. Researchers have investigated whether the academic programs or courses fulfill the requirements established by domain experts, regardless of by manual or automatic methods. We separate these works into two groups based on what types of information they aim to extract.

The first group of research [59, 60, 45] focused on how the course content distributes on a predefined set of knowledge categories. For instance, Bain et al. [59] manually scrutinized textbooks and counted the pages spent on the knowledge categories in the domain of accounting information system. A statistical model was adopted in [45] to predict the distribution of computer science courses over some predefined knowledge categories. These works looked at the composition of course content rather than the coverage of a knowledge category, which is the main difference with our work.

The second group of research attempted to understand whether and to what extent a knowledge category is covered by academic programs or individual courses. For example, Lennox and Diggens [33] interviewed the school staffs on whether their curricula touch on the ideal knowledge summarized by domain experts. Contractor et al. [32] tackled the problem of detecting the most related knowledge category for a given piece of course materials in an automatic manner. Both of these works only evaluated whether the knowledge category of interest is covered or not. Other research took a further step to inspect the extent to which knowledge categories are covered. For instance, Macdonald and Fougere [34] used 5-point Likert scale to review how a textbook covers the categories about the subject of software piracy. Ishihata et al. [35] conducted a survey on how the informational science and engineering departments cover the core knowledge categories in this domain. They obtained the teaching hour of each department spending on each category from the questionnaire and then divided it by the required hour of the category to compute the coverage. What they achieved is close to our goal in this study, however, we address the problem in an semi-automatic way by processing the texts of courses and knowledge categories. Lastly, Kawintiranon et al. [36] utilized information retrieval techniques to estimate how a course is associated with a knowledge category. The association score they extracted is actually the ratio of how many keywords in the knowledge category also appears in the course content. In this sense, it is similar to the concept of coverage in our study. However, their association score gets larger when the keyword appears more frequent in the course content. As a result, their association score does not strictly fall into the range of [0, 1], which is different from what we attempt to estimate in this study.

3.2.2 Graph-based document relatedness estimation

Relatedness (or similarity) of documents is an important metric in information retrieval and it has received continuous attention. One stream of research in this area utilized the knowledge graph to represent a document, thus the relatedness can be captured from the graphical perspective. It would seem that our work falls in a branch in this stream of research. However, our work is independent from those works for two reasons:

- What we aim to estimate, as called "coverage", is distinct from "relatedness". The relatedness of a document d_1 and another document d_2 derives from the related and the unrelated information of d_1 and d_2 (Refer to [61] for a detailed clarification.). Thus, if any of d_1 and d_2 contains more unique information, the relatedness become less. In contrast, the coverage of d_1 by d_2 is decided by the common information of d_1 and d_2, and all the information of d_1. In other words, no matter how much unique information d_2 contains, the coverage remains unchanged unless d_1 is unchanged.

- Theoretically, the coverage of d_1 by d_2 can be approximated by computing the relatedness of d_1 and $d_1 \cap d_2$ (common information of d_1 and d_2). However, previous methods [62, 63, 64] lacked the quantification of the total information contained in a document, which is essential for estimating coverage. For example, Schuhmacher and Ponzetto [62] estimated the relatedness of d_1 and d_2 by inverting the cost of converting the graph of d_1 to the graph of d_2. However, this approach only captures what is unrelated (the cost to edit the differences of two graphs) but not what is related (the identical part of two graphs). Thus, it is unsuitable to estimate coverage in this type of approach.

3.2.3 Centrality in text processing

Centrality has been applied in text processing tasks mainly for document summarization [65, 66, 67, 68] and other tasks such as keyword extraction [69, 66], topic identification [70], and term weighting [71] etc.

Some of these works used centrality score as a feature in further computation. For example, Xie [69] utilized centrality measures as the features in a supervised model—decision tree to predict the noun phrases that should be included in the abstract of a document. Rousseau and Vazirgiannis [71] adopted centrality scores as term weights to represent a document, which was then used to retrieve the proper document for a query. Other works used the centrality score directly to select important sentences/words to represent a document [65, 70, 66, 67, 68]. All of these works utilized degree centrality, which results in a sentence being considered important if it has a larger number of direct neighbors. In this study, we value the indirect connections between vertices as well. Therefore, we adopt another type of centrality and it will be further explained in Section 3.4.3.

The construction of the graph used to compute centrality plays a key role in applying centrality in such tasks. Some of the works [69, 71, 66] added edges based on the co-occurrences of sentences or phrases. This is built upon the assumption that a sentence can represent a document better if it appears together with more sentences. While in other works [65, 66, 67], an edge indicates two sentences are similar to each other. The meaning of edges was defined more specifically in [70] and [68]. Coursey and Mihalcea [70] modeled the relationship between two phrases if one is mentioned in the document of the other one. Rashidghalam et al. [68] adopted the relationships (e.g., derive, is-a, part-of, and related etc.) existing in the BabelNet ontology as the meanings of edges in their graph. Our definition of the edge is closer to the ones in the last two works and the details will be explained in Section 3.4.2.

3.3 Problem Formalization

3.3.1 Knowledge category

Domain knowledge categorization is used as a reference to manage knowledge. With the diverse backgrounds of MOOCs, a standard domain knowledge categorization becomes especially helpful. According to our preliminary survey, there exist curriculum

Table 3.1: A part of the knowledge categories in CS2013.

KA	***KU***	***Topic***
Information Management	Information Management Concepts	· Information systems as socio-technical systems...
	Database Systems	· Approaches to and evolution of database systems...
	Data Modeling	· Data modeling...
	Indexing	· The impact of indices on query performance...
	Relational Databases	· Mapping conceptual schema to a relational schema...
	Query Languages	· Overview of database languages...
	Transaction Processing	· Transactions...
	Distributed Databases	· Distributed DBMS...
	Physical Database Design	· Storage and file structure...
	Data Mining	· Use of data mining...
	Information Storage and Retrieval	· Digital libraries...
	Multimedia Systems	· Standards (e.g., audio, graphics, video)...

guidelines which attempt to categorize the knowledge that an academic program should include. Some examples are, "Curriculum Guidelines for Undergraduate Programs in Statistical Science" (by American Statistical Association)[10], "ASM Curriculum Guidelines for Undergraduate Microbiology" (by American Society for Microbiology)[11],

[10]`http://www.amstat.org/asa/education/Curriculum-Guidelines-for-Undergraduate-Programs-in-Statistical-Science.aspx`, accessed June 11, 2019.

[11]`https://www.asm.org/index.php/guidelines/curriculum-guidelines`, accessed June 11, 2019.

"Computer Science Curricula 2013" (by ACM/IEEE-CS) [41], etc. Among these existing knowledge categorizations, we select "Computer Science Curricula 2013" (henceforth, CS2013) as an instance of the knowledge categorization for the reasons that: *a*) it covers a wide range of knowledge in the domain and organizes it into a category structure with more than one level; and *b*) the authors are more familiar with the domain of computer science. In CS2013, the knowledge is dubbed as *Topics*, and then grouped into *Knowledge Units* (*KUs*) and *Knowledge Areas* (*KAs*). Table 3.1 shows the structure and some instances of knowledge categories in CS2013.

3.3.2 Problem definition

In this study, we adopt the term "concept" to refer to a technical term, denoted as c. Since we use the course syllabus as a textual representation of the course content, we denote a course as s to avoid a duplicate notation with concept. Besides, we denote a knowledge category as k. Both s and k are defined as a set of concepts. That is to say, given a syllabus s and a knowledge category k, we aim to estimate the ratio that the concepts required in k are covered by s, which is denoted as $cov(k|s)$.

3.4 Methodology

3.4.1 Intuition

Intuitively, the coverage of k by s can be captured in Equation (3.1). With the denominator being the total knowledge that is required in k and the numerator being the knowledge that is both required in k and taught in s, the result provides us a ratio that can be comprehended as the knowledge coverage of k by s. Then, our goal is to estimate the two items in Equation (3.1), namely, the required knowledge and the required and taught knowledge.

$$cov(k|s) = \frac{\text{The knowledge required in } k \text{ and also taught in } s}{\text{The knowledge required in } k} \tag{3.1}$$

Since we have already constrained k and s to the form of a concept set, we need to quantify how important the concepts are to the whole set. Suppose we have a concept set $C_1 = \{$"Relational Model","Transaction Processing","Concurrency Control"$\}$ and a concept set $C_2 = \{$"Relational Model","SQL","Relational Algebra"$\}$. Although the

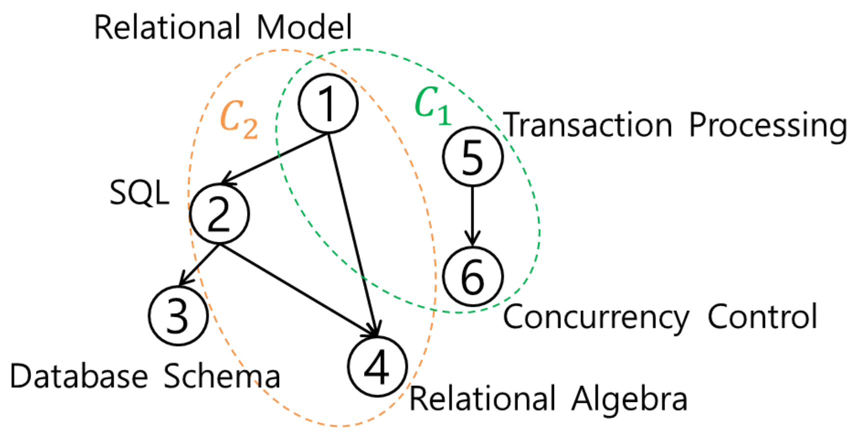


Figure 3.2: An example of the taxonomy of concepts.

concept “Relational Model” is required by both C_1 and C_2, it is not equally important to these sets. This is the underlying relationship among concepts that makes them behave differently when they are combined with different concepts. In this study, we model the concepts and their relationships as a taxonomy. Figure 3.2 demonstrates an example of the taxonomy of some database-related concepts. An edge indicates that the concept being pointed is a part of the other concept. As we can see, both “Relational Model” and “Transaction Processing” are important and relatively independent concepts in the domain of database. Therefore, it is likely that C_1 requires broader and shallower knowledge of “Relational Model”. While in C_2, “SQL” and “Relational Algebra”, two sub-concepts of “Relational Model” are also required, which indicates more concentrated and deeper knowledge of “Relational Model” is required.

Based on the above intuition, we propose a method consisting of three steps— I) Taxonomy Construction, II) Concept Importance Computation, and III) Concept Importance Aggregation. Firstly, we construct a taxonomy which embeds the relationships among concepts. Then, we utilize the idea of centrality in the taxonomy to compute the importance of concepts to a concept set. When the concept importance is computed, we then simply aggregate the importance values of the concepts that are contained in k or s as the amount of required knowledge or taught knowledge, respectively. Figure 3.3 depicts the overall framework of this study and we will explain the main method by steps in the following sections.

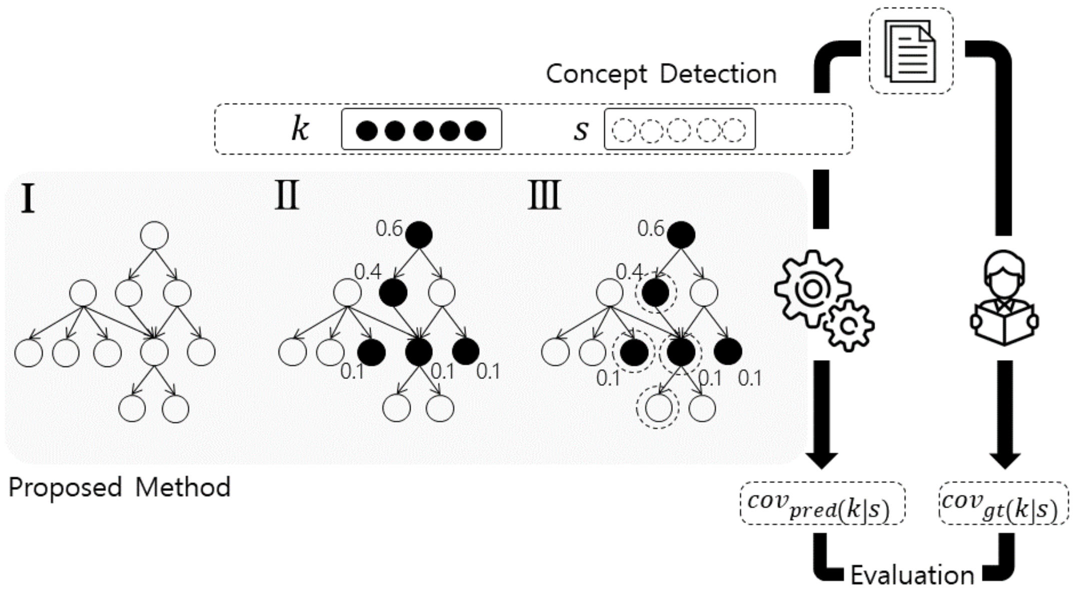


Figure 3.3: The overall framework of this study. The white circles represent concepts and the directed edges indicate relationships between concepts. To separate the concepts appearing in different types of documents, filled circles are used for knowledge categories and dotted circles used for syllabi. $cov_{pred}(k|s)$ is the knowledge coverage of k by s estimated by our proposed method while $cov_{gt}(k|s)$ is the one assigned by domain experts.

3.4.2 Taxonomy construction

This step corresponds to step I in Figure 3.3, in which a taxonomy of concepts is constructed. We denote the taxonomy as a directed acyclic graph $G =< V, E >$, where V is a set of concepts and $E = \{(c_i, c_j)|\text{if learning } c_j \text{ is necessary to understand } c_i\}$ is a set of directed edges. As mentioned in the example in Figure 3.2, there should be an edge from “Relational Model” to “SQL”, since it is inevitable to learn the knowledge of “SQL” to understand “Relational Model”. We define the edge this way deliberately for the computation of concept importance and the reason will become clearer in the next section.

The quality of the taxonomy plays a significant role in the computation of the concept importance. Therefore, we are cautious to construct a taxonomy with reliable edges. Based on our definition of the edge, we consider the textbook is a valuable recourse to extract the relationships between two concepts. If we treat the chapter title as the concept to be explained, then the concepts being mentioned in the chapter can be treated as the necessary concepts in understanding the chapter title. When given a set of concepts and a set of textbooks, we propose Algorithm 1 to establish edges for those concepts.

```
Algorithm 1: Establish edges from textbooks
   Input: $V$: set of $c$ that is given, $B = \{b_1, b_2, \cdots, b_n\}$: set of textbooks
   Output: $E_1, E_2, \cdots, E_n$
1  $E_1, E_2, \cdots, E_n \longleftarrow \emptyset$
2  for $i \longleftarrow 1$ to $n$ do
3      for $c \in V$ do
4          if $c$ shows in the index of $b_i$ then
5              $P_c \longleftarrow$ the pages $p$ where $c$ appears
6              for $p \in P_c$ do
7                  $T \longleftarrow$ the titles $t$ where $p$ appears and the corresponding levels
                     of the titles $l_t$.   // $l_t$ is the level of $t$ in the table
                     of content of the textbook.  A lower level has a
                     greater $l_t$ value
8                  sort $T$ based on $l_t$ in descending order
9                  for $t \in T$ do
10                     if $t$ shows as $c^{'} \in C$ then
11                         $E_i \longleftarrow E_i \bigcup \{(c^{'}, c)\}$
12                         break
```

In Algorithm 1, for every concept in the initial set, if it appears as a terminology in the index of the textbook, we then check whether the titles of the chapters where it appears are also concepts in the initial set. If so, an edge from the chapter concept to the index concept is established. Line 7 is a guarantee that when there exist multiple levels of chapters of a page, only the strongest edge (from the chapter concept at the lowest level) is included.

3.4.3 Concept importance computation

This step corresponds to step II in Figure 3.3, where we firstly introduce two ways to compute the importance of the concepts and then put forward a method to combine the two ways.

Uniform Computation Method

A naive way to compute the importance of the concepts in a set is to treat them uniformly important. We denote the importance of a concept c in a knowledge category k computed by the uniform computation method as $Imp_U(c|k)$, and it is generated as $Imp_U(c|k) = 1$.

Centrality-based Computation Method

As we have discussed before, the concepts in a set are not uniformly important since they represent knowledge of different depths and widths. For instance, in the concept set $C = \{$“Relational Model”, “SQL”,“Relational Algebra”$\}$, the knowledge of “SQL” and “Relational Algebra” contributes to the understanding of “Relational Model”. Thus, it is possible that C requires profound knowledge of “Relational Model” with special interests in the knowledge of “SQL” and “Relational Algebra”. How can we differentiate the importance of concepts? Recall our definition of the edges in the taxonomy— a concept has an edge to another concept if it can be better understood by learning the other concept. Therefore, we consider the concepts that have more access to other concepts in the taxonomy are more important in a set. This attribute is called “centrality” and used to find out the most “central” member in the context of social network analysis. The meaning of “central” depends on how it is defined in specific applications and different graph properties are used to compute centrality.

Based on our assumption that the importance of a concept is decided by the extend to which it could be understood by learning other concepts in the set, more important concepts should have more and intimate access to other concepts. Closeness centrality [72] serves our need in the sense that it treats a vertex as more central if it is closer to all the other vertices. A shorter total distance indicates more direct paths towards other concepts, thus, better understood and more important.

The original closeness centrality. A general equation used to compute the closeness centrality of v, $C(v)$, is shown in Equation (3.2), where $dis(v,u)$ is the length of the shortest path from v to u, and $n-1$ is the possible smallest total distance of v to other vertices. Thus, $C(v)$ presents the inverse average distance of v towards other vertices, which is normalized to the range $[0, 1]$.

$$\begin{aligned} C(v) &= \left(\frac{\sum_{u \neq v} dis(v,u)}{n-1}\right)^{-1} \\ &= \frac{n-1}{\sum_{u \neq v} dis(v,u)} \end{aligned} \tag{3.2}$$

Dealing with disconnected graphs. The original equation for computing closeness centrality is meaningless if the graph is not connected. Since all the total distance of a vertex towards other vertices becomes infinity even if there is only one vertex not connected to any vertices. This results in all the vertices having a centrality of zero, which underestimates the importance of the connected vertices. In the following, we explain some variants [73, 74, 75, 76] to deal with disconnected graphs and how we choose the proper one to solve our problem.

(a) Large-value-replaced closeness centrality [73]. In this model, the distance to unreachable vertices are replaced by a large value instead of infinity. In Equation (3.3), m in the second item in the denominator is the number of unreachable vertices of v and β is a parameter to modify this value (which is commonly set to the diameter of the graph). Note that the first item in the denominator only counts the distance to reachable vertices of v.

$$C_{LV}(v) = \frac{n-1}{\sum_{u \neq v} dis(v,u) + m\beta} \tag{3.3}$$

(b) Harmonic closeness centrality [74, 75]. As shown in Equation (3.4), this model computes the centrality of a vertex by summing up its inverse shortest distance to other vertices, which is then normalized by the possible maximum total inverse distance $(n-1)$. When there is no path from v to u, $dis(v,u) = \infty$, which results in a zero in the summation.

$$C_H(v) = \frac{1}{n-1} \sum_{u \neq v} \frac{1}{dis(v,u)} \tag{3.4}$$

(c) Components-based closeness centrality [76]. In this model, the centrality of vertices are computed independently inside each connected components, and then normalized by the relative size of this component to the whole graph.

Table 3.2: An example of using variants of closeness centrality for disconnected graphs.

Concept	v	$C_{LV}(\beta = 6)$	C_H
"Relational Model"	1	0.31	0.50
"SQL"	2	0.25	0.40
"Database Shema"	3	0.17	0.00
"Relational Algebra"	4	0.17	0.00
"Transaction Processing"	5	0.20	0.20
"Concurrency Control"	6	0.17	0.00

However, in each connected component, this model cannot cope with directed graphs that may still have disconnected pairs of vertices. For this reason, we exclude this variant from the candidate models to compute concept centrality.

Table 3.2 shows the closeness centrality values of the exemplar taxonomy (in Figure 3.2) by using different models explained in (a) and (b). As we can see the distributions of two models, large-value-replaced closeness centrality tends to generate closer values for all vertices. In this example, we set β to the possible smallest large value, namely the number of vertices in the graph, as the replacement of infinity. If we enlarge the value of β, the values of all the vertices will get closer and closer, which is not desirable in our problem setting. On the contrary, harmonic closeness centrality succeeds to differentiate more important vertices (i.e., 1, 2, and 5) and less important vertices (i.e., 3, 4 etc.). Therefore, we adopt harmonic closeness centrality as our centrality-based method to compute the importance of concepts in a set.

Combined Computation Method

Since our taxonomy is directed, the leaf vertices will be underestimated during centrality computation (see the last column in Table 3.2). Although we consider leaf concepts are less important than the concepts in the upper levels, they should not be ignored completely. Thus, it is effective to combine the uniform computation method and centrality-based computation method to achieve a balanced importance of the concepts. We introduce a parameter to modify the trade-off between the differentiation of concept importance and the preservation of importance of less important concepts. To sum up, the importance of a concept to a set is computed by

$$Imp(c|k) = \alpha \cdot C_H(c|(k,G)) + (1-\alpha) \cdot Imp_U(c|k)\,, \tag{3.5a}$$

$$C_H(c|(k,G)) = \frac{1}{|k|-1} \sum_{c' \in k \setminus c} \frac{1}{dis(c,c')}. \tag{3.5b}$$

Note that, when computing $dis(c,c')$, it may involve other concepts contained in the whole taxonomy G but not in k. When $\alpha = 0$, it is identical to use uniform computation method, which is considered as our baseline. We tuned the value of α from $[0,1]$ in the experiment and found that the best performance appears when α is valued between $[0.85, 0.95]$.

3.4.4 Concept importance aggregation

In step II, we have computed the importance of a concept to a concept set. On one hand, the required knowledge of k can be obtained by summing up the importance values of its concepts. On the other hand, the required but also taught knowledge is simplified as the sum of the importance values of concepts that appear both in k and s. The ultimate equation used to compute the knowledge coverage of k by s is

$$cov(k|s) = \frac{\sum_{c \in k \cap s} Imp(c|k)}{\sum_{c \in k} Imp(c|k)}\,. \tag{3.6}$$

This step is notated as step III in Figure 3.3.

3.5 Experiment

3.5.1 Dataset

There are 18 *KAs* in CS2013 and we only pick one *KA* to test the effectiveness of our proposed method. The reasons are:

- It is non-trivial to generate a taxonomy of domain knowledge correctly and automatically. To make sure the emphasis of this study falls on the step of computing concept importance, we adopt a balanced strategy to generate a reliable taxonomy which only allows us to implement it on a limited number of *KAs*.
- It is costy to generate ground truth for the dataset. We resort to domain experts to assign the knowledge coverage of *KUs* by courses. This requires the domain

Table 3.3: Statistics of the documents of k and s.

	k	s
Number of documents	12	26
Average word count of the documents	50.42	253.96

experts being considerably familiar with a domain. It is practically hard for us to reach domain experts across the extensive scope of computer science.

Among the 18 *KAs* defined in CS2013, we then chose "Information Management" as our preliminary dataset. Firstly, the authors are more familiar with this *KA*, which leads to a more insightful result analysis. Secondly, information management is viewed as a microcosm of computer science [77] and we think it is proper to choose a representative *KA* in this domain to try out our proposed method. Then, we collected 26 syllabi of courses that are related to information management. Among them, 7 courses are online courses and 19 are courses being provided in brick-and-mortar universities. Table 3.3 gives the basic information of the documents of k and s.

Regarding the ground truth, we asked two domain experts[12] to assign the knowledge coverage of all the pairs of k and s after reading the documents of 12 *KUs* and 26 courses. In detail, they were required to follow these instructions:

1. Read through the documents of 12 *KUs* to make sure you understand what knowledge is required. It may be helpful to form an image and keep in mind of what sub-topics you will teach and how much time you will spend in order to teach the required knowledge.

2. Read the course syllabi one by one and assign the percentage values while referring to your comprehension of the required knowledge. For the syllabi without explicit indications on how much time is spent on each topic in it, you may judge from the overall content of the course and estimate the volume of its content by treating it as a regular one-semester course.

3. Adjust the coverage values you have assigned to make sure they are judged under the same criterion whenever necessary.

[12]One of them is the author of this paper, and the other one is not.

Table 3.4: Statistics of the Wikification results of k and s.

	k	s
Average number of concepts	25.83	98.54
Total number of concepts	264	1436

Table 3.5: Evaluation on the Wikification results.

KU	# of unrelated concepts	Average centrality of unrelated concepts
IM01	6	0.000
IM03	2	0.000
IM04	8	0.024
IM06	1	0.000
IM07	3	0.000
IM10	4	0.000

The correlation coefficient of the coverage values collected from two experts is 0.855 ($p < 0.0005$). Therefore, We consider their assignment as reliable and then took the average coverage values as the final ground truth.

3.5.2 Concept detection

Our proposed method is based on the assumption that the knowledge categories and course syllabi are given as sets of concepts. Therefore, the concept detection is a pre-processing of the documents we have collected. Specifically, we adopt several existing tools to detect the concepts that are defined in the knowledge base Wikipedia. Wikipedia is an online encyclopedia that can be edited and updated by massive users. It has become a valuable knowledge resource for tasks in various fields such as information retrieval, knowledge engineering, and natural language processing, to name a few. For a given document, Wikification is a process in which the phrases and their corresponding Wikipedia articles are detected and extracted. In this study, we treat the titles of Wikipedia articles as concepts and adopt four Wikifiers to convert documents of k and *s* into concept sets. Among the four Wikifiers, one is our original tool[13] and the other three were developed in previous research [78, 56, 79]. We then took the union of the detected concepts by all the four Wikifiers as the final concept sets.[14] We accept the detected Wikipedia articles as the given sets of concepts. No special process is conducted to find potentially missing concepts since it is beyond the scope of this study. Table 3.4 reports the numbers of concepts detected in the process of Wikification.

There remains a concern that including multiple Wikifiers may increase the noisy concepts that are not actually related to this document. However, we expect the negative effect of these concepts can be alleviated when projecting them on the taxonomy. To verify this, we randomly selected six of the *KUs* and asked two evaluators (both of them are PhD candidates and one is the first author.) to check whether the detected Wikipedia articles are related to the document or not. Three levels were used to rate a Wikipedia article— related, somewhat related, and not related. We then computed the average centrality values of the Wikipedia articles that are considered as not related by at least one of the evaluators. Table 3.5 reports that the noisy Wikipedia articles get rather low centrality values, which implies that the centrality computation succeeds to suppress the importance of wrongly detected Wikipedia articles.

3.5.3 Taxonomy construction

In the experiment, we set the union of the concepts detected in all the knowledge categories as V, and followed Algorithm 1 to extract $E = E_1 \cup E_2 \cup E_3$ manually from three classic textbooks [77, 80, 81] in the domain of information management. Note that we include all the edges that can be found in at least one of the textbooks, since the number of valid edges decreases dramatically if we raise the threshold to two. We also removed one edge that causes a cycle in the taxonomy.[15] Consequently, we obtained a taxonomy of 264 vertices and 245 edges.

[13] We first use NLTK package to extract noun phrases from the documents. Then, the noun phrases are used as query to search related Wikipedia articles in the Bing search engine.

[14] In preliminary experiments, we tried to use the number of wikifiers by which a concept is detected as an indicator of the reliability of the concept. However, the performance didn't improve significantly. Therefore, we do not include this factor in this study.

[15] One cycle (Transaction processing$\rightleftharpoons$Concurrency control) is found in our dataset. Since coping with the cycles is not essential in this study, we simply removed the edge from Concurrency control to Transaction processing to avoid the cycle.

3.5.4 Evaluation framework

In this study, our main concern is how the coverage values can help a user to make an informed decision of learning. Therefore, the predicted $cov(k|s)$ value should be as close to the ground truth as possible. We evaluate the result in two scales.

Evaluating all pairs

In this evaluation scale, we require every individual $cov(k|s)$ being comparable to each other. That is to say, we evaluate whether the $cov(k|s)$ values of all pairs of *KUs* and courses are in consistent with the ones of the ground truth. Two metrics are adopted:

- **Pearson Correlation Coefficient (Pearson, thereafter)** [82] is used to reflect whether the predicted coverage values is "propotional" to the ones of the ground truth. We denote the set of knowledge categories and the set of syllabi as K and S, respectively. The ground truth of the coverage of the i^{th} k by the j^{th} s is denoted as $cov_{gt}^{(i,j)}$ and the prediction of our proposed method as $cov_{pred}^{(i,j)}$. Then, Pearson value is computed as

$$P = \frac{\sum(cov_{gt}^{(i,j)} - cov_{gt})(cov_{pred}^{(i,j)} - cov_{pred})}{\sqrt{\sum(cov_{gt}^{(i,j)} - cov_{gt})^2 \sum(cov_{pred}^{(i,j)} - cov_{pred})^2}}, \tag{3.7}$$

  where cov_{gt} is the average value of $cov_{gt}^{(i,j)}$, and cov_{pred} is the average value of $cov_{pred}^{(i,j)}$.

- **Mean Squared Error (MSE, thereafter)** is used to check whether the predicted coverage values have a small deviation from the ground truth. As shown in the following equation, the errors that have a larger difference from the ground truth get larger penalties.

$$MSE = \frac{\sum(cov_{pred}^{(i,j)} - cov_{gt}^{(i,j)})^2}{|K||S|} \tag{3.8}$$

Evaluating by course

In this evaluation scale, we focus on the coverage estimation inside every course. This is driven by the consideration that syllabi may be written in different styles or levels of detailedness even the courses cover a *KU* to a similar degree. We treat a syllabus

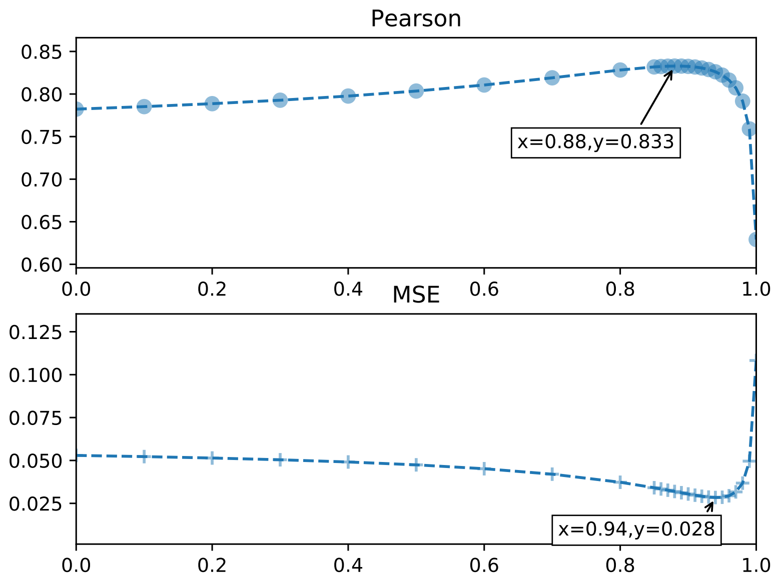


Figure 3.4: The results of evaluating all pairs in scatter plots. The x axis represents the values of α and the y axis represent Pearson and MSE values, respectively. The y value in the box is the maximum Pearson or minimum MSE value, and x value is its corresponding α value.

as a vector in the space of its knowledge coverage with the KUs, which is denoted as $\overrightarrow{cov_s(K)} =< cov(k_1|s), \ldots, cov(k_n|s) >$, where $n = |K|$. Then we compute the cosine similarity between the predicted $\overrightarrow{cov_s(K)}$ and the ground truth.

3.5.5 Results

Evaluating all pairs

The only parameter in our method is α, which indicates the extent to which the harmonic closeness centrality is utilized in the computation of concept importance. Figure 3.4 is the scatter plot of the α values and their corresponding Pearson and MSE values. $\alpha = 0$ represents the baseline experiment in which uniform importance values of concepts in k and s are used to compute the coverage. When inspecting the α values other than zero, both Pearson and MSE values reach a peak at some relatively high value and then drop dramatically when α equals 1. In detail, Pearson reaches its peak when α is valued of 0.88, and MSE reaches its peak when α is valued of 0.93. Overall, the method performs

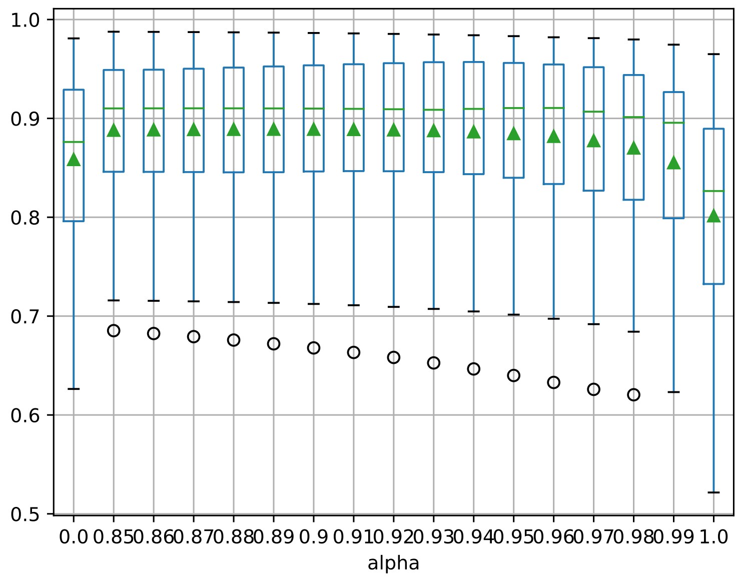


Figure 3.5: The result of evaluating by course. The x axis represents the values of α and the y axis represents the cosine similarity values.

best on both evaluation metrics when α is valued in the range $[0.88, 0.93]$. This proves that the idea of using centrality to compute the importance of a concept is valid in the estimation of knowledge category coverage. Moreover, the combination of uniform computation method and centrality-based computation method plays a significant role in applying the idea of centrality to solving our problem.

Evaluating by course

In Figure 3.5, we plot the box-plots of the cosine similarity values of 26 courses for the experiments with α valued in $[0, 0.85 - 0.99, 1]$. As can be observed, the mean value (represented in green triangles in the figure) of cosine similarity has an obvious rise from the baseline method ($\alpha = 0$), starts to fall from where $\alpha = 0.94$, and finally drops when $\alpha = 1$. Similar trends can be found on median, the first-quartile, the third-quartile, the minimum and the maximum values of cosine similarity. We conclude that our proposed method is valid to estimate the knowledge category coverage for a course when choosing the appropriate parameter to combine the uniform and centrality-based computation methods.

3.6 Discussion

In this section, we investigate on the result in more depth. Specifically, we check over the most important concepts of each *KU* and the performance on different *KUs*. In Table 3.6, we list the five most important concepts with their importance values when $\alpha = 0.88$. Regarding Table 3.7, we treat a *KU* as a vector in the space of its knowledge coverage by the syllabi and then compute the cosine similarity of the predicted one and the ground truth.

First of all, we find the most important concepts are representative of the *KUs*. A quick verification is to check whether we can infer the main topic of the *KU* merely from the important concepts without knowing the title of this *KU*. As shown in Table 3.6, it is easy to judge that *KU5* is about relational database design and *KU7* requires the knowledge of transaction processing.

On the other hand, we find that our method is weak to the knowledge categories that contain "isolated" or "general" concepts. For example, the proposed method (when $\alpha \neq 0$) is not working for *KU12* (see Table 3.7). A potential reason is that this *KU* contains relatively new and inter-disciplinary concepts that are underpresent in classic textbooks. Thus, these concepts get low importance values since they are "isolated" in the taxonomy. Another example is the comparison of *KU1* and *KU10*. Both of the *KUs* contain a very limited number of important concepts (i.e., Database, Computer data storage in *KU1* and Data mining in *KU10*). However, if we compare the performance on these two *KUs* in Table 3.7, it can be seen that utilizing centrality-based computation method more is decreasing the accuracy of *KU1* while increasing the accuracy (when α is in the range of $[0, 0.8]$) of *KU10*. We analyze this is caused by the different generality of concepts in the domain. That is to say, Database and Computer data storage tend to appear in various *KUs* and they are not that important to *KU1*, while Data mining is unique to *KU10* and it deserves to be highly valued. Although our method is able to differentiate the importance of a concept in different concept sets, further consideration on how to modify the importance values of a "general" concept is needed.

3.7 Summary

In this study, we firstly define the knowledge coverage of a knowledge category by a course as the extent to which the knowledge required in the category is also taught in the course. Then, we propose a centrality-based computation method to estimate the

Table 3.6: The five most important concepts of *KUs* ($\alpha = 0.88$).

KU1: Information Management Concepts		*KU2*: Database Systems		*KU3*: Data Modeling	
Database	0.208	Database management system	0.419	Relational database	0.306
Computer data storage	0.148	Database	0.403	XML	0.296
Information	0.120	Database transaction	0.183	Data model	0.237
Sociotechnical system	0.120	Transaction processing	0.167	Data modeling	0.237
Information retrieval	0.120	Relational database management system	0.151	Semi structured data	0.237
KU4: Indexing		*KU5*: Relational Databases		*KU6*: Query Languages	
Computer data storage	0.289	Relational model	0.513	SQL	0.503
Database	0.234	Database design	0.472	Database	0.495
SQL	0.169	Database normalization	0.419	Relational database	0.417
Database index	0.149	Relational database	0.325	Query language	0.291
Index database	0.149	Data integrity	0.261	Stored procedure	0.169
KU7: Transaction Processing		*KU8*: Distributed Databases		*KU9*: Physical Database Design	
Database transaction	0.462	Database	0.384	Computer data storage	0.357
Transaction processing	0.413	Distributed database	0.294	Database index	0.323
Computer data storage	0.364	Parallel database	0.266	Index (database)	0.323
Concurrency control	0.218	Query optimization	0.221	Database	0.289
Data buffer	0.120	Computer data storage	0.180	Database design	0.120
KU10: Data Mining		*KU11*: Information Storage and Retrieval		*KU12*: Multimedia Systems	
Data mining	0.309	Information storage and retrieval	0.153	Information retrieval	0.120
Data	0.120	Document management system	0.136	Digital library	0.120
Interactive visualization	0.120	Inverted index	0.136	User interface	0.120
Cluster analysis	0.120	Information retrieval	0.133	Data compression	0.120
Algorithm	0.120	Information security	0.133	Multimedia	0.120

Table 3.7: The result of evaluating by *KUs*.

α	*KU1*	*KU2*	*KU3*	*KU4*	*KU5*	*KU6*	*KU7*	*KU8*	*KU9*	*KU10*	*KU11*	*KU12*
0.0	0.753	0.907	0.971	0.808	0.953	0.945	0.926	0.887	0.885	0.657	0.793	0.627
0.1	0.753	0.907	0.972	0.809	0.953	0.946	0.927	0.887	0.886	0.657	0.793	0.627
0.2	0.753	0.908	0.972	0.809	0.954	0.948	0.929	0.886	0.887	0.657	0.793	0.627
0.3	0.752	0.908	0.972	0.809	0.955	0.950	0.930	0.886	0.888	0.658	0.793	0.627
0.4	0.752	0.908	0.972	0.810	0.956	0.952	0.931	0.885	0.890	0.659	0.794	0.627
0.5	0.751	0.908	0.972	0.810	0.957	0.955	0.933	0.884	0.891	0.660	0.794	0.627
0.6	0.750	0.909	0.972	0.811	0.959	0.958	0.935	0.883	0.893	0.660	0.794	0.627
0.7	0.749	0.909	0.972	0.812	0.960	0.961	0.936	0.880	0.895	0.661	0.794	0.627
0.8	0.746	0.910	0.972	0.814	0.962	0.966	0.938	0.875	0.898	0.661	0.794	0.627
0.9	0.736	0.910	0.971	0.815	0.962	0.970	0.937	0.862	0.898	0.647	0.795	0.627
1.0	0.647	0.895	0.956	0.803	0.956	0.972	0.928	0.806	0.875	0.411	0.726	0.000

In each column, the cell is colored based on which range the difference of its value with the one of the baseline method ($\alpha = 0$) falls in.

−1.000	−0.100	−0.075	−0.050	−0.025	−0.010	0.000	+0.010	+0.025	+0.050	+0.075	+0.100	+1.000

concept importance to the knowledge categories, which is then aggregated to estimate the knowledge coverage. The experiment has shown that our method can generate closer knowledge coverage values to the ground truth assigned by human experts, compared to the uniform computation method.

Some future challenges remain. We only experiment on one *KA* in this study, and we expect to extend it to other areas in the domain of computer science once we have upgraded the technique to build a broader taxonomy. In the method, we did not carry out any technique to deal with general concepts that appears in multiple concept sets. It is an interesting task to consider how to modify the importance values of a concept to different sets based on what other concepts are contained in the sets.

CHAPTER 4

COURSE ORDERING

In Chapters 2 and 3, we focus on learning goals presented in domain knowledge categories, which are closer to the world of academia. In this chapter, we put emphasis on other learning goals which relate to job opportunities. Generally, performing tasks in the industries requires a compound mastery of various knowledge and skills. As a result, we need to take more courses into consideration and balance the trade-offs among those courses. In this chapter, we address the course ordering problem for a given job-oriented goal while considering the prerequisite relationships between courses.

4.1 Introduction

Many college students experience the following when their graduations are approaching: flooded with a large amount of job postings, closely looking at their curriculum vitae (CV), and struggling to attract recruiters' attention among a bunch of competitors' applications. With the development of open and online education, we hold a vision that, in the near future, the students will be able to freely construct their own curriculum, which is not prescribed by the institutions. In addition, students are encouraged to accumulate working experience earlier as it is helpful for building the professional identity [11]. Consequently, it is necessary for students to keep adapting their learning experience to achieve fruitful learning and successful career development according to frequent updates in the job market.

Table 4.1: Example of courses.

ID	Courses
c_1	Computer Programming 1
c_2	Computer Programming 2
c_4	Data Structures and Algorithm Analysis
c_6	Web Site Design and Development
c_{12}	Advanced Web Design and Programming (. . . Students examine advanced topics in Hyper Text Markup Language, **Cascade Style Sheet** and **JavaScript** for . . .)
c_{13}	Database Systems
c_{17}	Programming Methods (. . . Students learn a combination of visual programming using C# and scripting language using **Python** in this course. . . .)
c_{21}	Web-based Information Systems (. . . Students use a variety of web development tools and programming/**scripting** languages. . . .)

"Technical terminologies" are important building blocks of job opportunities. For instance, `scripting language` is one of such technical terminologies frequently required in IT job positions. In our work, we address technical terminologies as students' learning goals. Given `scripting language`, what is the best order to take courses for students? It is a difficult question even if the number of candidate courses is small. Table 4.1 lists an example of eight courses related to `scripting language`. According to the course title and the snippet of the course content, we find that courses c_{12} and c_{17} are helpful for learning some scripting languages such as Cascade Style Sheet, JavaScript, and Python. In addition, course c_{21} also addresses some scripting languages though they are not the main topics of the course. As a result, we recommend courses c_{12}, c_{17}, and c_{21} to students, in which priority is given to c_{12} and c_{17}. However, some of these courses are built on the basis of other courses. For example, before a student learns advanced knowledge about web design and programming in course c_{12}, the student should understand the basic knowledge of web design and development in course c_6. Given a course, some courses that students need to learn prior to it are called the *prerequisites* of it. Therefore, we need to not only prioritize the courses related to the target terminology but also pay attention to the prerequisite relationships among courses when providing an order to take them. Actually, the number of candidate courses and the complexity of the course prerequisite relationships are expected to be much larger than the example shown in Table 4.1, which motivates this study.

Many researchers have been working on recommending learning materials. They took into consideration various learning needs and constraints such as completion time of the courses is shortened, course requirements are met, important knowledge is included [37, 20, 38, 39], and so on. However, these works have not explored the order among multiple related courses in a prerequisite-based course network. Other works aimed at recommending job opportunities to students [83, 84, 85, 86] or recommending learning materials for career demands [22, 25]. While the former works ignored the course prerequisite relationships and orders [83, 84, 85, 86], the latter ones focused on short-term learning scenarios in which only one course was recommended [22, 25]. In contrast, our work is helpful for planning long-term learning in which multiple courses are involved.

We develop a two-step approach to solve the course ordering problem: The first step estimates the relatedness of the courses to the technical terminology (hereafter, course-terminology relatedness). Then the second step determines the order of the courses based on the estimated relatedness and prerequisite relationships. In these two steps, we address the first step as a general task of relatedness estimation and put emphasis on the second step. Specifically, we then propose a method for ordering courses based on Markov decision process and conduct comparative experiments.

Furthermore, we explore whether the order is relevant from other pedagogical perspectives such as whether the very basic course is ranked first, the course is close to its prerequisites in the order, and so on. Experimental results show our proposed method can prioritize the related courses and place them closer to their prerequisites. However, one of the comparative methods demonstrates a similar performance as the proposed method. We then discuss the difference between our proposed method and the comparative method in detail.

The contributions of our work are summarized as follows:

- To the best of our knowledge, our work is the first that attempts to order the courses towards a job-oriented learning goal by following the prerequisite constraint.

- Our work is information retrieval-driven and education-aware one. In other words, we mainly serve the users whose information gain is maximized by following the order, and also explore whether the generated order is helpful for students or not from educational perspectives.

- We construct a fair-scale dataset annotated with three kinds of labels that denote relatedness between courses and technical terminologies.

4.2 Related work

4.2.1 Recommending learning materials

Recommending learning materials has attracted a lot of attention in educational research communities. These works took into account a variety of elements such as prerequisite relationships, difficulty, popularity, availability, and importance of the learning materials to meet different learning needs [37, 20, 38, 39]. In the following, we summarize their works and highlight the differences between our work and theirs.

Parameswaran and Garcia-Molina [37] addressed the problem of recommending items with prerequisite relationships. They aimed at finding the set of k items which has the maximum total score and meanwhile meets the prerequisite constraint. Therefore, the order of the k items does not matter if the total score is maximized. However, their algorithm can be adapted to solving our problem and the details are explained in Section 4.4.2. Xu et al. [20] addressed more complex course constraints such as prerequisite relationships, course availability, and mandatory/elective requirements. They developed a forward-search backward-induction algorithm to optimize course sequences with shorter time needed for graduation. In other words, they aimed to minimize the time of a course sequence if all the mandatory and enough elective courses are included. Unlike these works, our work determines the detailed order of related courses rather than whether related courses are included in the order or not.

Zhu et al. [38] and Shi et al. [39] recommended learning paths from a knowledge map by meeting multiple constraints such as whether the paths contain unlearnt, important or popular knowledge nodes, and so on. In their problem settings, the start and end knowledge nodes are given and all the paths connecting the two nodes are treated individually. In our work, multiple related courses exist in a graph (i.e., multiple end nodes) and the order of nodes on multiple paths should be considered.

4.2.2 Connecting academia and industries

To fill the gap between the academia and the industries, many works attempted to recommend job opportunities to students or recommend learning materials for career demands.

A common approach can be observed in a category of research works in which the most "similar" job to the student's educational background is recommended [83, 84, 85, 86]. Commonly, job postings and student CVs were utilized, and then various techniques were adopted such as ontology-based skill similarity [83], recurrent neural network [84], latent Dirichlet allocation [85], and naïve Bayes classifier [86] to estimate the similarity between them. These works treated the students' learning experience as a whole, namely, all the courses are completed and the students' acquisition level of those courses influences the recommendation performance. On the other hand, our work puts more emphasis on making a plan for learning, in which the course prerequisite relationships and orders are more important.

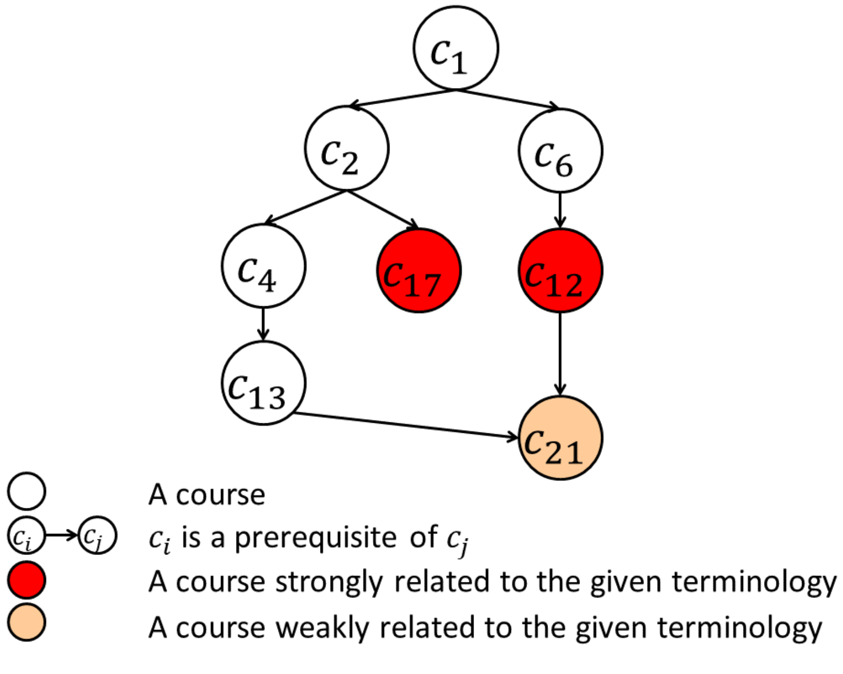
Another category of research works attempted to identify relevant courses or training program for job positions or demands of career development [22, 25]. Srivastava et al. [22] inferred a course which a student is most likely to take next by mining from a large-scale course enrollment data. Wang et al. [25] recommended a course based on the employee's current competencies modelled from their skill profiles. For this reason, their works concentrated on the short-term learning needs, namely, the next one course is recommended for the student. Our work not only extracts the most necessary courses but also achieves prerequisite-aware course ordering for long-term learning needs.

4.3 Task Formulation

4.3.1 Awareness of prerequisite relationships

Generally, we need to acquire some knowledge before understanding more advanced knowledge. For instance, if we do not know "algorithms" at all, it is difficult to understand the "complexity of an algorithm". In taking courses, following the prerequisite relationships among courses is essential.

In traditional educational institutions, course prerequisite relationships are defined by the curriculum designers. In open and online education, massive learning materials are available. For the latter learning environment, research works focusing on the extraction of prerequisite relationships among concepts [87, 88, 89, 90] are helpful. Works on prerequisite extraction are orthogonal to our work and we assume the prerequisite relationships are given in our problem setting.



c_1
c_2
c_6
c_4
c_{17}
c_{12}
c_{13}
c_{21}
A course
c_i is a prerequisite of c_j
A course strongly related to the given terminology
A course weakly related to the given terminology

Figure 4.1: Prerequisite relationships among courses.

4.3.2 Definition of a "relevant" order

Given a target technical terminology, we address how to effectively acquire the knowledge related to it. Figure 4.1 shows prerequisite relationships among the courses in Table 4.1 and how these courses are related to `scripting language`. Then, the relevant order is generated by highly ranking the courses which are related to the terminology. In other words, $c_1 \rightarrow c_6 \rightarrow c_{12} \rightarrow c_2 \rightarrow c_{17} \rightarrow c_4 \rightarrow c_{13} \rightarrow c_{21}$ (hereafter, Order 1) is a relevant order as all the related courses (i.e., c_{12}, c_{17}, and c_{21}) are ranked in the highest positions where they could appear. The goal of our work is to identify the most relevant order by taking prerequisite constraint into account.

4.3.3 Observation from pedagogical perspectives

Depending on the complexity of the course network and the number of related courses, there may exist more than one relevant order of courses for a technical terminology. Even for the simple course relationships in Figure 4.1, we can find another order, $c_1 \rightarrow c_2 \rightarrow c_{17} \rightarrow c_6 \rightarrow c_{12} \rightarrow c_4 \rightarrow c_{13} \rightarrow c_{21}$ (hereafter, Order 2), which is equally effective as Order 1 in terms of prioritizing related courses. However, Orders 1 and 2 give different effects on a student's learning experience. For example, Order 1 places c_{12} prior to c_{17}, which is desirable if the student prefers learning a basic course first and then other advanced courses. In contrast, Order 2, where c_{12} is placed closer to c_{21}, is desirable if a student dislikes the time lag between a course and its prerequisites. Hence, we explore the effectiveness of an order from the following pedagogical perspectives.

- **Specific fundamentality.** Specific fundamentality defines how likely a course will form the basis towards understanding the technical terminology. Prioritizing such courses makes a student's basic knowledge solid, helping the learning of more advanced knowledge.
- **General fundamentality.** In contrast to specific fundamentality above, general fundamentality defines how likely a course will form the basis towards understanding the whole domain. A general fundamentality-focused order is especially important in a learning context with high uncertainty. In other words, if students change their goals during the learning process, they can still benefit from the courses they have already completed since the courses also contribute to the understanding of other terminologies.
- **Local reference.** Local reference refers to placing the prerequisites of a course closer to it. From a cognitive point of view, to shorten the time lag between courses with dependency helps reduce the extra cognitive load in learning [91, 52].

Theoretically, our concerns are various aspects that affect a student's learning experience. However, it is difficult to satisfy all types of students' learning preferences at the same time as some of them are inherently contradictory to each other. Our work proposes a method for identifying "relevant" orders and explores whether the orders are effective from the aforementioned perspectives.

4.3.4 Problem definitions

A course set V and the prerequisite relationships $E = \{(c_i, c_j) \mid c_i, c_j \in V, c_i \text{ is a prerequisite of } c_j\}$ forms a course graph $G =< V, E >$. In our work, when given a technical terminology t and a course graph G, we aim at identifying an order $Ord(V|t) = (c_i, \ldots, c_j)$ subject to the prerequisite relationships and prioritizing the courses closely related to t. In the following sections, we denote $pos(c, Ord)$ and $Ord[i]$ as the position of c in Ord and the course at the i^{th} position in Ord, respectively.

4.4 Methodology

To acquire the knowledge of a given technical terminology, we need to not only identify the related courses but also pay attention to the prerequisites when deciding the order to take them. Intuitively, our task consists of the following two steps:

(STP1) Estimation of how each course is related to the given technical terminology, namely, course-terminology relatedness; and

(STP2) Ordering the courses based on their course-terminology relatedness and prerequisite relationships.

Course-terminology relatedness can be estimated by matching the textual information of courses and terminologies. To this end, STP1 is a general task in the domain of information retrieval and a variety of existing approaches can help realize it [92]. We then propose a method for ordering of how to take courses after estimating course-terminology relatedness.

4.4.1 Proposed method—Markov Decision Process-Based Ordering (MDPOrd)

Intuition of adapting Markov decision process

Markov decision process (MDP) is a stochastic sequential decision model, which has been applied to a wide range of problems such as inventory management, equipment maintenance, communication systems, and so on [93]. The main idea behind this model is to find the best set of decisions that optimizes the long-term goal.

Given a technical terminology, ordering of courses can be viewed as a process in which we start with the courses without any prerequisites and keep adding them until we reach the ones that are highly related to the terminology. In this context, how to select a course in a higher position depends on (1) how likely the course will be related to the terminology itself and (2) how likely the course will ultimately lead to a highly-related course in the graph. As a result, every step in the course ordering process should be based on the expectation of the future gain by taking that step. We employ MDP as it can model our task well and identify an order of courses that optimizes the information gain for the technical terminology.

Formulation of MDPOrd

A general model of MDP consists of decision epochs T, states S, actions A, rewards R, transition probabilities P, discount factor γ, and policies Π, namely, denoted as $\{T, S, A, R, P, \gamma, \Pi\}$. In this framework, a decision maker follows a policy to take an action at each epoch and also receives a reward and transition between states. The main goal is to find a policy which leads to a state with optimal discounted future gain. We explain how each components are modeled in the following:

Decision Epochs. Let T be the discrete time steps in the system. In our work, we adopt infinite horizon $T \equiv \{1, 2, \ldots\}$ to find the best policy, which assumes the system does not know how long the process will last at any time step. Infinite horizon assures a stationary policy for any state regardless at what time step it reaches the state. The practical meaning of infinite horizon in our work can be explained as follows: the system will suggest the same course order whenever the students start learning. This is reasonable if we assume the students always have sufficient time to learn the courses.

States. Let S be the set of possible states the system occupies at a time step. In our work, a state s is a set of courses a student has completed. As our work needs to follow prerequisite relationships, for any state s, if it includes a course c, it must include the prerequisites of c. Therefore, we denote a possible $s = \{c_j \in V \mid \exists(c_i, c_j) \in E \implies c_i \in s\}$. It is time-consuming to exhaust all the possible states, especially, a course is allowed to have more than one prerequisite in the graph. Table 4.2 demonstrates the process of generating valid states for the course graph in Figure 4.1 and the followings detail the process:

1. We add a dummy node to the graph and connect it to the original root nodes (the value of its in-degree is 0) in the graph. The addition of a dummy node assures that
   (a) all the sub graphs are linked as one graph if more than one root exist, and
   (b) the generated states will always include an empty set, which is the start state to find an order of courses.
2. We traverse the graph in a depth-first way from the dummy node. Whenever we explore a new node, we generate new states by adding this node to the current states based on the following two rules:

Table 4.2: Process of generating valid states for the course graph in Figure 4.1.

Step	**Node**	**From**	**Parents visited**	**Generating states**
1	*dummy*	-	True	$\emptyset$
2	c_1	*dummy*	True	$\{c_1\}$
3	c_2	c_1	True	$\{c_1, c_2\}$
4	c_4	c_2	True	$\{c_1, c_2, c_4\}$
5	c_{13}	c_4	True	$\{c_1, c_2, c_4, c_{13}\}$
6	c_{21}	c_{13}	False	-
7	c_{17}	c_2	True	$\{c_1, c_2, c_{17}\}$ $\{c_1, c_2, c_4, c_{17}\}$ $\{c_1, c_2, c_4, c_{13}, c_{17}\}$
8	c_6	c_1	True	$\{c_1, c_6\}$ $\{c_1, c_2, c_6\}$ $\{c_1, c_2, c_4, c_6\}$ $\{c_1, c_2, c_4, c_{13}, c_6\}$ $\{c_1, c_2, c_{17}, c_6\}$ $\{c_1, c_2, c_4, c_{17}, c_6\}$ $\{c_1, c_2, c_4, c_{13}, c_{17}, c_6\}$
9	c_{12}	c_6	True	$\{c_1, c_6, c_{12}\}$ $\{c_1, c_2, c_6, c_{12}\}$ $\{c_1, c_2, c_4, c_6, c_{12}\}$ $\{c_1, c_2, c_4, c_{13}, c_6, c_{12}\}$ $\{c_1, c_2, c_{17}, c_6, c_{12}\}$ $\{c_1, c_2, c_4, c_{17}, c_6, c_{12}\}$ $\{c_1, c_2, c_4, c_{13}, c_{17}, c_6, c_{12}\}$
10	c_{21}	c_{12}	True	$\{c_1, c_2, c_4, c_{13}, c_6, c_{12}, c_{21}\}$ $\{c_1, c_2, c_4, c_{13}, c_{17}, c_6, c_{12}, c_{21}\}$

(RL1) Given a node at exploration, new states must be generated from the states generated from its parent and other explored descendants. For example, when we are exploring c_{17}, its parent c_2 and other descendants of c_2— c_4 and c_{13} are already explored. We generate new states by adding c_{17} to all the generated states from c_2, c_4, and c_{13} (Steps 3 to 5).

(RL2) Given nodes with multiple parents, we only generate new states at the last time we explore it, and new states must contain all of its parents. For example, when we first explore c_{21} in Step 6 where the other parent c_{12} has not been explored yet, it is skipped. The last time we explore c_{21} when all of its parents have been explored, we can generate new states by adding c_{21} to the states in Step 9 which include both of its parents c_{12} and c_{13}.

Finally, we take the union of all the states generated alongside the traverse as S.

Actions. A denotes the union of actions that the decision maker is allowed to take at each state. In this case, an action is the behavior to take a new course at the current state. Subject to the constraint, the valid action for a state is any course whose prerequisites (if any exists) are already included in the state. For example, in Figure 4.1, given $s = \{c_1, c_2\}$, the valid actions are c_4, c_{17}, and c_6.

Rewards. $R = S \times A$ denotes the immediate reward obtained from taking a course c at state s. In our work, the reward comes from the information gain towards understanding the technical terminology by taking a course. To this end, we use course-terminology relatedness $rel(c,t)$ to represent the immediate reward. Equation (4.1) defines how an immediate reward $r(s,c)$ is computed from taking c at s:

$$r(s,c) = \begin{cases} rel(c,t) & c \notin s, s \cup \{c\} \in S \\ 0 & otherwise \end{cases} . \tag{4.1}$$

Transition Probabilities. $P = S \times A \times S$ denotes the distributions of the probabilities that the system transits from states to states by taking courses. $p(s,c,s')$ denotes the probability that the system transits from s to s' by taking c. Note that $\sum_{s'} p(s,c,s') = 1$. In our work, we simplify the transition probability by considering the system will keep transiting to the next state if some courses are taken. If the action is invalid, the state remains unchanged. Equation (4.2) defines this:

$$p(s,c,s') = \begin{cases} 1 & s \cup \{c\} = s' \\ 0 & otherwise \end{cases} . \tag{4.2}$$

Discount Factor. γ denotes a parameter ranged in $0 \leq \gamma < 1$ modifying how much of the future gain values currently. This parameter is mainly set for a mathematical reason, allowing the decision maker to find an optimal policy. In our experiment, we set γ to 0.96, which is commonly used value in MDP.

Policy. $\Pi = S \rightarrow A$ denotes the moving pattern of a decision maker at each state. In MDP model, a policy that maximizes the expected value of all the states is defined as an optimal policy π^*. The expected value of a state $ExpVal(s)$ comes from the immediate reward of taking an action and the expected value of the next state. The best we can expect from a state $ExpVal^*(s)$ is the maximum value of all policies and the optimal policy for s is the action that maximizes $ExpVal(s)$, defined by Equations (4.3) and (4.4), respectively:

$$ExpVal^*(s) = \max_c (r(s,c) + \gamma \sum_{s'} pr(s,c,s')ExpVal(s')) \,, \tag{4.3}$$

$$\pi^*(s) = \arg\max_c (r(s,c) + \gamma \sum_{s'} pr(s,c,s')ExpVal(s')). \tag{4.4}$$

Equations (4.3) and (4.4) can be solved using dynamic programming algorithms which find an approximation of the optimal policy until it converges.

Ordering. In this step, we utilize the output of MDP model to provide relevant course ordering. Once the optimal policy π^* is found, we start with the empty set and follow the best actions to move from one state to another state. If all the courses have a positive reward value, following the optimal policy leads us to a state in which all courses are included. If not, the state remains unchanged at some point where no more reward can be obtained by taking a new course. In our work, we sort the remaining non-rewarded courses topologically and append them to the order.

4.4.2 Comparative methods

We compare MDPOrd with the following three different types of methods:

- Aggregated-Relatedness-Based Ordering (AggRelOrd),
- Personalized-PageRank-Based Topological Sorting (PageRankTS), and
- Greedy-Value-Pickings-Based Ordering (GVPickings).

The intuition behind AggRelOrd and PageRankTS is that the priority of a course in the order is determined by its direct relatedness to the terminology and how likely it will be the prerequisite of other related courses. On the other hand, GVPickings selects the courses as a set of paths, where a highly related course having many prerequisites still have a chance to be selected at an earlier stage. We compare these three methods to verify whether our proposed approach works better in ordering the courses. In the following sections, we explain each of three comparative methods.

Aggregated-Relatedness-Based Ordering (AggRelOrd)

As described above, a course should be prioritized as it is related to the terminology itself and it is the basis of other courses. This method simply treats the maximum relatedness among all the descendants of a course as an aggregated score to indicate its priority. Let $D(c)$ be the set of courses that have a path from c in the course graph, namely, the descendants of c in the graph. Let $aggRel$ denote the aggregated relatedness of a course. Then $aggRel$ is computed as:

$$aggRel(c|t, V) = (\max_{c' \in c \cup D(c)} rel(c', t)) + |D(c)|\delta, \tag{4.5}$$

where δ is a very small value added to assure that the score a course gets is always larger than its descendants, thus ranked higher in the order. At last, the courses are sorted in the order of their $aggRel$ scores.

Personalized-PageRank-Based Topological Sorting (PageRankTS)

PageRank [94] estimates the probability of a "random walker" who ultimately stops at each node in the network if it follows the links. Therefore, the probability distribution shows the linkage of the network. The more incoming links, the higher probability a node gets. Furthermore, a personalized PageRank [94, 95] enables the model to estimate a mixed probability distribution of following both the network linkage and a personalized preference over the nodes. As described at the beginning of Section 4.4.2, the position of a course in the order should be determined by the relatedness to the terminology and how likely the course will be the prerequisite of other related courses. Here, the direct relatedness between the course and the terminology can be represented in the personalized preference in the PageRank model. In addition, the likelihood of being a prerequisite of other related courses can be captured in the network links part in PageRank model.

We follow the matrix-vector notation in [95] to explain how the PageRank-based score of each course is computed. Let $\mathbf{v}$ be the PageRank scores over the courses in the graph, and $\mathbf{u}$ be the course-terminology relatedness. Let $\mathbf{M}$ be the transition matrix of the graph where $M_{ij} = \frac{1}{|indegree(c_j)|}$ if c_i is prerequisite of c_j and $M_{ij} = 0$ otherwise. $\beta \in [0, 1]$ is a teleportation constant and modifies the probability that the "random walker" follows the personalized preference over the nodes.[16] Solving Equation (4.6)

[16]We only report the results obtained by $\beta = 0.2$ in Section 4.5.4 as it gives the best performance.

gives a PageRank-based score **v** for each course.

$$\mathbf{v} = (1-\beta)\mathbf{M}\mathbf{v} + \beta\mathbf{u} \tag{4.6}$$

Algorithm 2: PageRank-based topological sorting (PageRankTS)

```
Input  : $G =< V, E >$: the course graph, $V$: the set of courses,
         $E = \{(c_i, c_j) \mid$ if $c_i$ is prerequisite of $c_j\}$: the set of course
         prerequisite relationships, $pr_i$: the PageRank-based score of $c_i$
Output : $Ord$: a queue of courses
1 $Ord \longleftarrow \emptyset$
2 while $G$ not empty do
3     $Root \longleftarrow$ the courses $c$ whose indegree is 0
4     Add $c$ to $Ord$ if $c \in Root$ and $c$ has the largest $pr$
5     $V \longleftarrow V \setminus c$
6     $E' \longleftarrow \{(c, c') \mid (c, c') \in E\}$
7     $E \longleftarrow E'$
```

Note that the score provided by the PageRank model is not necessarily subject to our prerequisite constraint. In other words, it is possible that a course which has many outgoing edges gets a higher score than its parent course who has no other children. To solve this problem, we rely on topological sorting to reorder the courses based on their PageRank-based scores. As shown in Algorithm 2, line 2-6 is the basic process of topological sorting with a modification in line 4, where the course with the highest PageRank-based score and no any prerequisites is always selected first.

Greedy-Value-Pickings-Based Ordering (GVPickings)

As described in Section 4.2.1, Parameswaran and Garcia-Molina [37] proposed Greedy Value Pickings which recommends the best set of k items with the maximum total score and meets the prerequisite constraint. However, their work did not consider the order of items in the set if the total score is maximized. While Greedy Value Pickings was not designed for ordering the items at the first place, we adapt it to our work. We enhance Greedy Value Pickings by utilizing the order that k items are added to the final set as the order required in our work, which is shown in Algorithm 3. The basic idea behind this algorithm is to always add the path of courses with the largest average

Algorithm 3: Greedy-Value-Pickings-Based Ordering (GVPickings)

```
Input  : $G =< V, E >$: the course graph, $V$: the set of courses,
         $E = \{(c_i, c_j) \mid$ if $c_i$ is prerequisite of $c_j\}$: the set of course
         prerequisite relationships
Output : $Ord$: the queue of courses
1  $Ord \longleftarrow \emptyset$
2  while $G$ not empty do
3      for $c \in V$ do
4          $Anc(c) \longleftarrow$ ancestors of $c$
5          $Anc'(c) \longleftarrow Anc(c) \cup \{c\}$
6          $value(c) = \sum_{a \in Anc'(c)} rel(a) / |Anc'(c)|$
7      $\hat{c} \longleftarrow \arg\max_{c \in V} value(c)$
8      Add a topological order of $Anc'(\hat{c})$ to $Ord$
9      $V \longleftarrow V \setminus Anc'(c)$
10     $E \longleftarrow E \setminus \{(u, v) \mid u \in Anc'(c), (u, v) \in E\}$
```

score to the order. As our course graph allows multiple parents for a node, all the paths towards a node should be included if we add a node (lines 4 and 5). Once a path (or paths) of courses is added to the final order, the average scores of the remaining paths are recomputed and the picking is conducted again. The process of computation and picking is repeated until all the courses are correctly ordered.

4.5 Experiment

4.5.1 Dataset

For the convenience of data collection and analysis, we select computer science domain to verify the effectiveness of our proposed methods.

Course graph

We collected the course syllabi and prerequisite relationships from the bachelor program of computer science in Thompson Rivers University.[17] We select this program as we can access sufficient course information and prerequisite relationships. As a result, we obtain 24 courses and 30 prerequisite relationships linked each other (see Table A.1 and Figure A.1 for further details).

Technical terminology

We extract a set of technical terminologies, which are frequently required by the job postings in a kaggle dataset.[18] The dataset consists of 19,000 job postings collected from the Armenian human resource portal CareerCenter during 2004 to 2015. We use 3,759 of the them which are labelled as IT-related. We first remove unnecessary sections such as "Company location", "Salary", "Application procedure", and so on by utilizing some basic text processing techniques. We then extract the technical terminologies from the pre-processed texts by applying entity extraction tool Wikifier, which works well in detecting abstractive concepts (e.g., "software development") and is linked to Wikipedia. More specifically, we employ TAGME [78] and DBpedia Spotlight [56] and include the union of the terminologies identified by any of them in the terminology set. While we address the job postings in IT-related domain only, the extracted terminologies still include general ones such as "knowledge","communication", and so on. As each of the terminologies has its own Wikipedia page, we filter out irrelevant terminologies that is greater than six hops distant from the Wikipedia category "Computing". Finally, we collect 3,803 terminologies and select the 100 most frequent terminologies in the job postings among them.

Ground truth

It is beyond the cognitive capacity even for an expert to decide the order of taking the courses with complex prerequisite relationships. In addition, it is difficult to integrate several orders into one. To avoid this, we first ask several experts what courses are necessary to take for a given technical terminology without directly collecting the

[17]`https://www.tru.ca/science/programs/compsci/programs/cs_bachelor_of_computing_science.html`, accessed January 18, 2021.

[18]`https://www.kaggle.com/madhab/jobposts/`, accessed January 18, 2021.

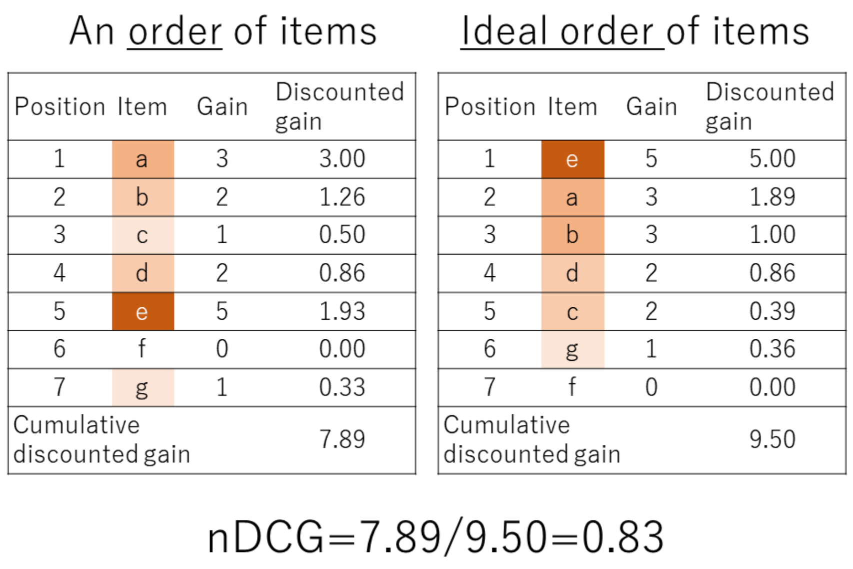
order of courses from them. We then evaluate the relatedness of an order with some traditional information retrieval metrics. We discuss the details in Section 4.5.3.

We invite five domain experts to construct our ground truth. Given a technical terminology, we first provide them with the graph and the syllabi of the courses, and then ask them to annotate which courses are necessary or preferable to take for better understanding the terminology. Note that "necessary" is a higher level of relatedness than "preferable". In this annotation task, each pair of a course and a terminology can be regarded as an item, and the domain experts give either of the following three labels: "necessary", "preferable", or "unnecessary" to the pair. Some technical terminologies such as `computer programming` and `productivity software` are too general or unrelated to the computer science domain. It is difficult for the domain experts to annotate a proper label for any of the courses for them. We call this type of terminology "irrelevant" and exclude the terminologies viewed as irrelevant by at least one domain expert. Finally, we obtain 67 terminologies in the dataset (see Table B.2 for further details).

We evaluate the agreement among five domain experts with Fleiss kappa coefficient [96], resulting in moderate agreement of 0.405. In a majority-vote method, the five domain experts and the three labels sometimes make it difficult to determine the final label. Therefore, we adopt DS algorithm [97], which is one of stochastic approaches, to estimate the probabilities of which label is likely to be given to the pair of course and terminology. Then, we choose the label with the highest probability as the ground truth.

4.5.2 Estimation of course-terminology relatedness

As described at the beginning of Section 4.4, our work does not mainly focus on estimating course-terminology relatedness. Using different techniques in relatedness estimation and then comparing their impact on the course ordering performance is beyond the scope of this work. Instead, we use the course-terminology relatedness annotated by the domain experts in STP1 to explore the best performance that STP2 can achieve. We discuss the details in Section 4.5.5. Here, we adopt a simple yet effective method, TF-IDF scheme [57] to estimate course-terminology relatedness in this work. More specifically, we use course syllabus and the leading section in Wikipedia page for the term as the course and terminology corpus. We first compute TF-IDF score from both of the corpus and then take the cosine similarity of the course and terminology



An order of items

Position	Item	Gain	Discounted gain
1	a	3	3.00
2	b	2	1.26
3	c	1	0.50
4	d	2	0.86
5	e	5	1.93
6	f	0	0.00
7	g	1	0.33
Cumulative discounted gain			7.89

Ideal order of items

Position	Item	Gain	Discounted gain
1	e	5	5.00
2	a	3	1.89
3	b	3	1.00
4	d	2	0.86
5	c	2	0.39
6	g	1	0.36
7	f	0	0.00
Cumulative discounted gain			9.50

nDCG=7.89/9.50=0.83

Figure 4.2: An example of how to compute nDCG.

vectors as the relatedness score.

4.5.3 Evaluation metrics

We adopt normalized discounted cumulative gain (nDCG) [98] to measure the relatedness, specific fundamentality, and general fundamentality of an order of courses. In addition, we propose a distance metric to measure the degree of local reference of an order of courses.

General model of nDCG

nDCG is a widely used metric in information retrieval, which measures how an order of items prioritizes the ones with a higher information gain. Figure 4.2 illustrates a simple example of how to compute nDCG. Firstly, the information gain of each item is discounted according to its position. The lower in the order, the more information gain is discounted. Secondly, the cumulative discounted gain is computed by summing up the discounted gain of all the items. At the same time, we reorder the items based on their information gain to obtain an "ideal" order of these items. Lastly, we normalize the cumulative discounted gain of the original order using the one of the ideal order.

nDCG meets our need to measure how an order prioritizes items that provide information of interest such as relatedness, specific fundamentality, and general fundamentality. Let $nDCG@k(Ord)$ be the normalized discounted cumulative gain of an order Ord for items at position k. $nDCG@k(Ord)$ is computed as follows:

$$nDCG@k(Ord) = \frac{DCG@k(Ord)}{DCG@k(Ord_{ideal})}, \tag{4.7}$$

$$DCG@k(Ord) = \sum_{i=1}^{k} \frac{gain(Ord[i])}{\log_2(i+1)}, \tag{4.8}$$

where $DCG@k(Ord)$ is the discounted cumulative gain of Ord at position k, Ord_{ideal} is the ideal order ranked according to the information gain of the items, and $gain(Ord[i])$ is the information gain of the i^{th} item in Ord. By replacing $gain(\cdot)$ with the information to be explored, we can adapt nDCG to measure the relatedness, specific fundamentality or general fundamentality of an order.

Relatedness of an order

We denote $Relatedness@k$ to measure how an order prioritizes related courses to the terminology. Let $rel^{gt}(c,t)$ be the score defined by the ground truth of course-terminology relatedness, which is transformed from the domain experts' annotated labels as follows:

$$rel^{gt}(c,t) = \begin{cases} \alpha & \text{if the label for } c \text{ is "necessary"} \\ 1-\alpha & \text{if the label for } c \text{ is "preferable"} \\ 0 & \text{if the label for } c \text{ is "unnecessary"} \end{cases}, \tag{4.9}$$

where α ($0 \leq \alpha \leq 1$) is a parameter to tune the level of relatedness.[19] Then $Relatedness@k$ is computed by replacing $gain(\cdot)$ in Equations (4.7) and (4.8) with $rel^{gt}(c,t)$.

Specific fundamentality of an order

We denote $SpecFdm@k$ to measure how an order prioritizes specifically fundamental courses to the terminology. Let $SpecFdm(c|t)$ be the specific fundamentality of a course for a terminology in the graph. We then define $SpecFdm(c|t)$ as follows:

$$SpecFdm(c|t) = \frac{\sum_{c' \in D(c)} rel^{gt}(c',t)}{\sum_{c' \in V \setminus c} rel^{gt}(c',t)}, \tag{4.10}$$

[19]We set $\alpha = 0.7$ in this work.

where $D(c)$ is the set of descendants of c in the graph. $SpecFdm(c|t)$ reflects the proportion of the related knowledge based on c to the total amount of related knowledge for the terminology. Then $SpecFdm@k$ is computed by replacing $gain(\cdot)$ in Equations (4.7) and (4.8) with $SpecFdm(c|t)$.

General fundamentality of an order

We denote $GenFdm@k$ to measure how an order prioritizes generally fundamental courses in the domain. Let $GenFdm(c)$ be the general fundamentatity of a course in the graph. Then it is computed as:

$$GenFdm(c) = \frac{|D(c)|}{|V| - 1}. \tag{4.11}$$

$GenFdm(c)$ reflects the proportion of the knowledge based on c to the total knowledge in the domain. Then $GenFdm@k$ is computed by replacing $gain(\cdot)$ in Equations (4.7) and (4.8) with $GenFdm(c)$.

Local reference of an order

We propose “Average Reference Distance (ARD)” to inversely measure how likely an order will place courses close to their prerequisites. We denote $ARD@k$ as the average distance between the courses and their prerequisites of an order at position k. Let $RelDis(c|Ord)$ be the reference distance of c in Ord. We then define $RelDis(c|Ord)$ as follows:

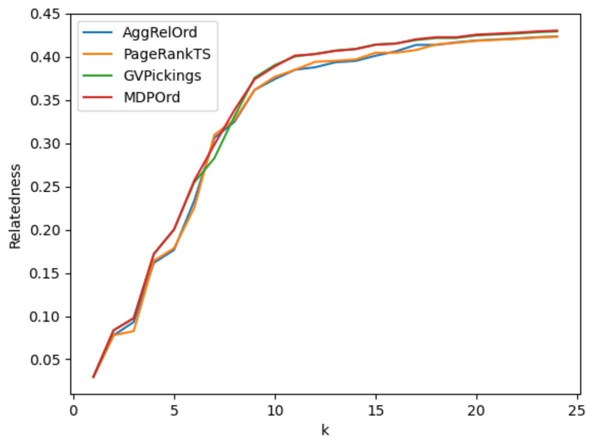
$$RelDis(c|Ord) = \begin{cases} 0 & Pre(c) = \emptyset \\ \frac{\sum_{c' \in Pre(c)} pos(c,Ord) - pos(c',Ord)}{|Pre(c)|} & \text{otherwise} \end{cases}, \tag{4.12}$$

where $Pre(c)$ is the set of direct prerequisites of c in $G =< V, E >$, i.e., $Pre(c) = \{c' \mid (c', c) \in E\}$. Then $ARD@k$ is computed as:

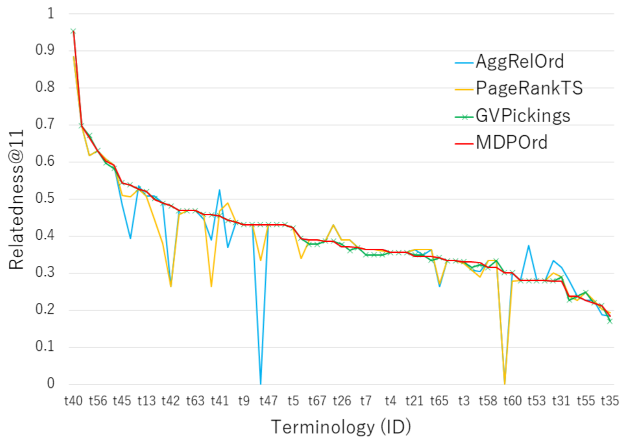
$$ARD@k = \frac{\sum_{i=1}^{k} RelDis(Ord[i])}{k - \sum_{i=1}^{k} \mathbb{1}\{Pre(Ord[i]) = \emptyset\}}. \tag{4.13}$$

Here, $\mathbb{1}\{\cdot\}$ is an indicator function which equals 1 if $\{\cdot\}$ is true, 0 otherwise.

In summary, $Relatedness@k$, $SpecFdm@k$, and $GenFdm@k$ range in $[0, 1]$. In addition, the larger the scores, the better the performance. On the other hand, $ARD@k$ ranges from 0 to some positive value based on the size and structure of a graph. The smaller $ARD@k$ is, the higher the degree of local reference, indicating a better performance.



(a) Average $Relatedness@k$, varying position k.



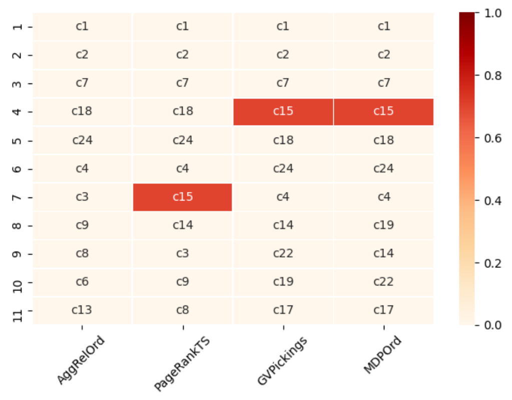
(b) $Relatedness@11$ over the terminologies. The terminologies are sorted in descending order based on MDPOrd scores.

Figure 4.3: Relatedness obtained by MDPOrd, AggRelOrd, PageRankTS, and GVPickings.

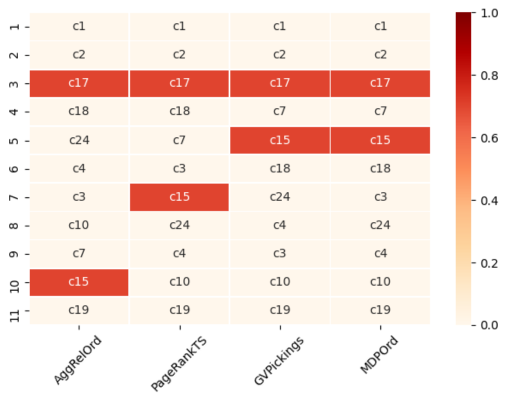
4.5.4 Results

Relatedness

Figure 4.3a shows the average nDCG-based relatedness of the orders obtained by the four methods (MDPOrd, AggRelOrd, PageRankTS, and GVPickings) described in Section 4.4 by varying k.



(a) `formal specification`. *Relatedness*@11 obtained by AggRelOrd, PageRankTS, GVPickings, and MDPOrd for this terminology is 0, 0.333, 0.431, 0.431, respectively.



(b) `debugging`. *Relatedness*@11 obtained by AggRelOrd, PageRankTS, GVPickings, and MDPOrd for this terminology is 0.483, 0.511, 0.543, 0.543, respectively.

Figure 4.4: Examples of orders generated by our proposed methods. In each subfigure, the orders for one terminology are presented. The row and cell color denote the position in the order and the course-term relatedness annotated by the domain experts, respectively. Only the top 11 courses in the orders are demonstrated to compactly show the results.

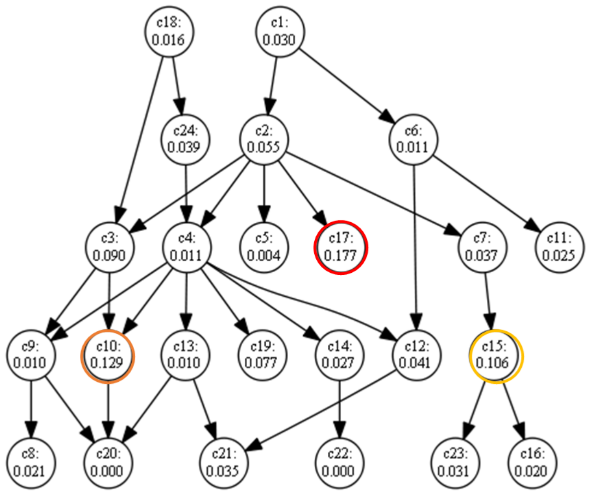
Overall, the *Relatedness* value ranges from 0 to 0.45 as the value of k increases. We observe that GVPickings and MPDOrd slightly outperform AggRelOrd and PageRankTS, especially when k is larger than 9. To further explore the performance over different terminologies, Figure 4.3b shows *Relatedness*@11 scores for each terminologies. According to Figure 4.3b, GVPickings and MDPOrd show a similar relatedness score distribution over the terminologies while AggRelOrd and PageRankTS show some significant performance drop for some of the terminologies.

What do these scores indicate in the orders? We pick two technical terminologies to demonstrate the relation between the *Relatedness* difference and the order difference. Figure 4.4 shows the orders generated by the four methods. In `formal specification`, c_{15} is considered necessary to take and it is not ordered within the top 11 courses generated by AggRelOrd, which leads to a *Relatedness* score of 0. Meanwhile, three lower positions of c_{15} (Software Engineering) in PageRankTS results in around 0.1 drop in the *Relatedness* compared with GVPickings and MDPOrd. In `debugging`, the domain experts annotate both c_{15} and c_{17} (Programming Methods) as necessary. All the methods successfully rank c_{17} in its optimal position while AggRelOrd and PageRankTS place c_{15} five and two lower positions than GVPickings and MDPOrd, respectively. The two lower positions for c_{15} result in a decrease of 0.03 in the $Relatedness$ of the order. The drop of 0.03 is small in the evaluation metric. However, the order forces the students to learn two more courses before they can reach the one which is helpful for the acquisition of knowledge on the terminology.

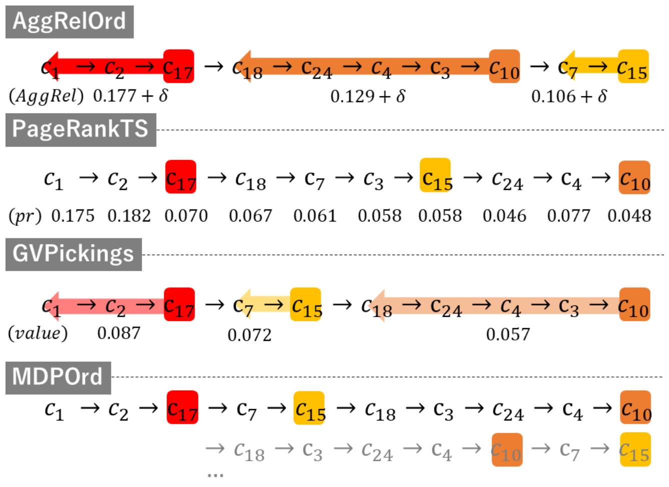
Comparison between the two terminologies indicates that the positioning of the first related course substantially influences the final $Relatedness$ score. While following related courses in the order do not give a large impact on the $Relatedness$ score, the wrong positioning of them results in extra learning cost in actual cases. In summary, we believe that GVPickings and MDPOrd work well.

To further discuss why GVPickings and MDPOrd work better than AggRelOrd and PageRankTS, we analyze the frameworks of four methods for the terminology `debugging`. Figure 4.5a shows the estimated course-terminology relatedness scores, in which c_{17} (Programming Methods), c_{10} (Operating Systems), and c_{15} (Software Engineering) give the top three scores 0.177, 0.129, and 0.106, respectively. However, only c_{17} (Programming Methods) and c_{15} (Software Engineering) are annotated as necessary by the domain experts. Thus, the relatedness of c_{10} (Operating Systems) is overestimated in STP1. Figure 4.5b illustrates how the order is determined in each method and how c_{10} affects the ordering process.

AggRelOrd aggregates the score of a course based on its most related descendant. As Figure 4.5b shows, c_1 and c_2 obtain their scores from the relatedness of c_{17}, namely, 0.177 with the corresponding small value δ. As a result, the order is determined by ranking the most related courses— c_{17}, c_{10}, and c_{15} with their ancestors, respectively. PageRankTS computes the score of each course from the scores of their children recursively. As shown in Figure 4.5b, the relatedness of c_{17}, c_{10}, and c_{15} is substantially transferred to other courses. Note that the PageRank scores of c_{15} and c_{10} are partially moved to their parents and partially comes from their children. In this case, c_{10} only has a non-related child c_{20}, resulting in a lower score (0.048) than the one of c_{15} (0.058). GVPickings picks paths based on their average relatedness. As shown in Figure 4.5b, the path from c_7 to c_{15} gets a higher average value (0.072) than the paths from c_{18} to c_{10} (0.057), resulting in lower rank for c_{10} while it has a higher relatedness than c_{15}. The decision maker in MDPOrd compares the expected values among multiple possible orders. For example, placing c_{10} prior to c_{15} results in an increase of expected value



(a) Course graph with the estimated course-terminology relatedness scores.


AggRelOrd
$c_1 \to c_2 \to c_{17} \to c_{18} \to c_{24} \to c_4 \to c_3 \to c_{10} \to c_7 \to c_{15}$
(AggRel) $0.177+\delta$ $0.129+\delta$ $0.106+\delta$
PageRankTS
$c_1 \to c_2 \to c_{17} \to c_{18} \to c_7 \to c_3 \to c_{15} \to c_{24} \to c_4 \to c_{10}$
(pr) 0.175 0.182 0.070 0.067 0.061 0.058 0.058 0.046 0.077 0.048
GVPickings
$c_1 \to c_2 \to c_{17} \to c_7 \to c_{15} \to c_{18} \to c_{24} \to c_4 \to c_3 \to c_{10}$
(value) 0.087 0.072 0.057
MDPOrd
$c_1 \to c_2 \to c_{17} \to c_7 \to c_{15} \to c_{18} \to c_3 \to c_{24} \to c_4 \to c_{10}$
$\to c_{18} \to c_3 \to c_{24} \to c_4 \to c_{10} \to c_7 \to c_{15}$
...

(b) Frameworks of the four methods determining the orders.

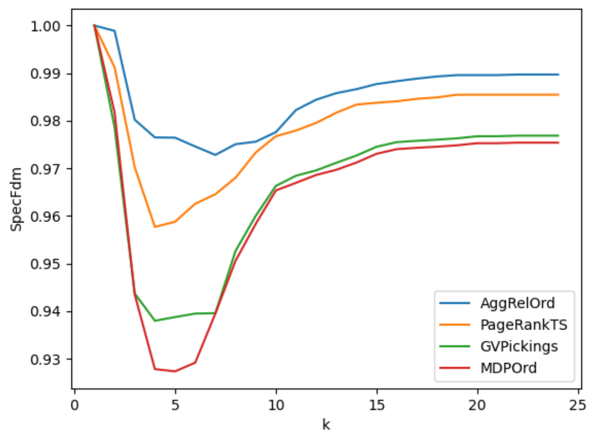
Figure 4.5: Analysis on the frameworks of AggRelOrd, PageRankTS, GVPickings, and MDPOrd for the terminology `debugging`.

from c_{10} but a larger decrease of that from c_{15}. Therefore, the upper order is finally selected.

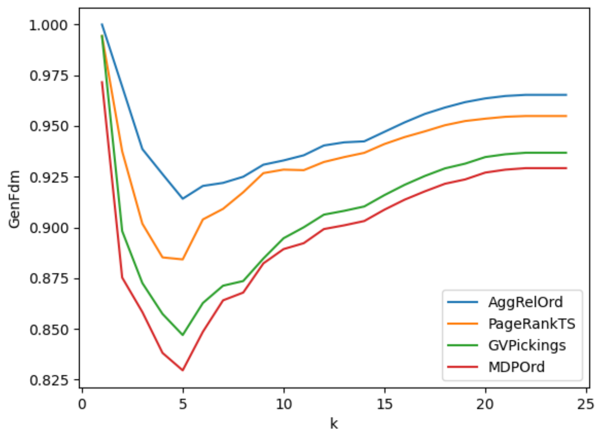
In summary, AggRelOrd work well in prioritizing a highly-related course no matter how deep it locates in the graph. Consequently, it cannot globally optimize relatedness of the order and wrongly estimated course often generates wrong order. PageRankTS does keep a balance between the relatedness of a course and the relatedness of its descendants. However, the courses with more children tend to be overestimated and the courses without any child tend to be underestimated. In contrast, GVPickings and MDPOrd consider the impact of selecting a course from a long-term perspective. Therefore, GVPickings and MDPOrd are more robust for wrongly estimated course-terminology relatedness and demonstrate more stable performances than AggRelOrd and PageRankTS.

Pedagogical metrics

Figures 4.6 and 4.7 show the $SpecFdm$, $GenFdm$, and ARD scores for the orders generated by the four methods, AggRelOrd, PageRankTS, GVPickings, and MDPOrd. We observe similar trends for $SpecFdm$ and $GenFdm$. Note that $SpecFdm$ and $GenFdm$ scores for the orders obtained by the four methods are relatively close to 1, which is due to the prerequisite relationships. In other words, a course that is prerequisite for more courses is usually ordered in a higher position, resulting in better fundamentality score. Among the four methods, AggRelOrd and PageRankTS outperform GVPickings and MDPOrd in both of the fundamentality scores, which can be inferred from the framework of AggRelOrd and PageRankTS. AggRelOrd chooses the maximum value among the course and its descendants as the aggregated value of this course. As a result, a course with more descendants, which is more fundamental one, is ordered higher in most cases. In PageRankTS, 80% of the priority of a course comes from the outgoing edges, which shows the fundamentality of the course. On the other hand, in ARD, we observe different trends as shown in Figure 4.7. The difference between (AggRelOrd, PageRankTS) and (GVPickings, MDPOrd) is getting larger as the value of k increases. GVPickings explores the courses in the unit of paths, which makes it easier to sequentially order a path of courses. In MDPOrd, if the first course of a path is not so closely related, the courses on the path tend to be ordered lower. In other words, the decision maker in MDPOrd tends to choose courses in the current path to starting a new path. In contrast, AggRelOrd and PageRankTS tend to prioritize courses with more descendants in the graph, resulting in more jumps in the order from one course to another without any prerequisite relationships.



(a) $SpecFdm$.



(b) $GenFdm$.

Figure 4.6: $SpecFdm$ and $GenFdm$ obtained by AggRelOrd, PageRankTS, GVPickings, and MDPOrd, varying position k.

4.5.5 Discussion

Analysis on performance obtained by STP1 and STP2

In this work, we put much emphasis on STP2 to identify an order of related courses for the technical terminology and adopt TF-IDF scheme to estimate course-terminology relatedness. As described in Section 4.5.2, we use the course-terminology relatedness annotated by the domain experts as the input of STP2 to explore the upper bounds of

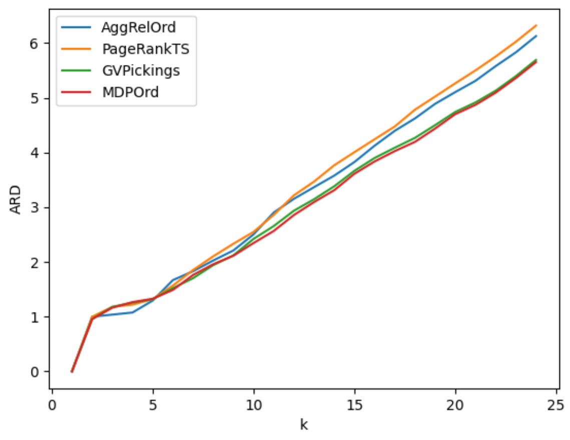


Figure 4.7: *ARD* obtained by AggRelOrd, PageRankTS, GVPickings, and MDPOrd, varying position k.

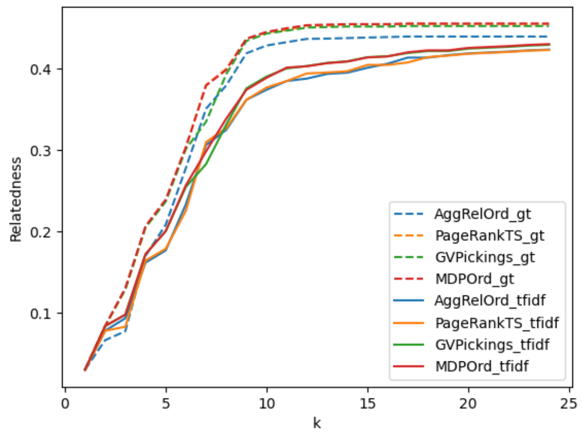
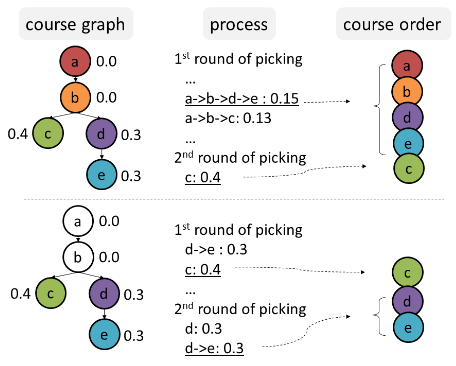


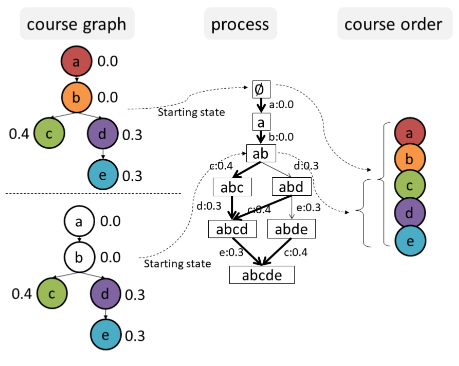
Figure 4.8: *Relatedness* obtained by AggRelOrd, PageRankTS, GVPickings, and MDPOrd by using ground truth and TF-IDF based relatedness in STP1, varying position k.

the performance of STP2.

Figure 4.8 shows that using the ground truth in STP1 results in a better *Relatedness* than using TF-IDF scheme-based course-terminology relatedness. As described in Section 4.5.4, the relatedness of `debugging` and c_{10} (Operating Systems) is overestimated by TF-IDF scheme. This is because the terms such as “system” and “memory” have a high frequency scores in c_{10}, and `debugging` contains these terms. However, if these



(a) GVPickings



(b) MDPOrd

Figure 4.9: Comparison between GVPickings and MDPOrd frameworks using a simple example.

terms are treated with more textual information, such as “operating system”, “main memory”, or “memory cache” in c_{10}, the relatedness with `debugging` is expected to be lowered. Therefore, improving the accuracy of relatedness estimation is one of future work.

Comparison between MDPOrd and GVPickings frameworks

As discussed in Section 4.5.4, MDPOrd and GVPickings show similar performances in all the metrics. However, they work in completely different ways, which gives impact on the utilization of their outputs and the refinement for more complex problem settings. Figure 4.9 illustrates a simple example of the difference, in which we use the same course graph and course-terminology relatedness values. At the top and bottom in Figure 4.9, we assume that the student has not completed any courses and completed course a and b, respectively. GVPickings and MDPOrd generate different course orders.

In GVPickings shown in Figure 4.9a, whenever it selects a path (or paths) with the highest average score, the value of each course on the path is discounted by the length of the path. In this example, the presence or absence of course a and b influences the order of course c, d, and e. This indicates that GVPickings requires the cost of taking the courses. GVPickings needs to be re-run for a new input of courses. To this end, GVPickings is not flexible to more complex problem settings.

On the other hand, MDPOrd works from a future-oriented perspective. The middle part in Figure 4.9b shows the state-action transition pattern in the MDP model, where a rectangle represents a state, an arrow represents an action, and a bold arrow indicates the best policy for a state. As long as the input set of courses is a valid state, the best policy (i.e., the order of taking courses) can be found in the same model. This characteristic derives from Markov property, which assumes that, given a current state, the probability of transition to a new state is independent of the past. In other words, the same model can be used to deal with different input of courses, resulting in a limitation that it ignores the cost of taking courses.

In other words, the frameworks to identify the best order in GVPickings and MDPOrd are different though they demonstrate similar performances in our work. MDPOrd is more flexible and promising to address more complicated tasks such as personalized course ordering.

4.6 Summary

In this work, we have addressed the problem of ordering the related courses for a given technical terminology while following the prerequisite relationships among courses. We also evaluated the effectiveness in the orders from pedagogical perspectives such as specific/general fundamentality, and local reference. We observed that our proposed method works well in prioritizing related courses and ordering them with a shorter reference distance.

In future work, we plan to address the followings: We used courses collected from one curriculum in which prerequisite information is available. Thus, we plan to explore whether we can apply our method to order courses in much larger learning resources such as Massive Open Online Courses (MOOC). In addition, we only evaluated the orders generated by our methods from a pedagogical point of view. It would be more useful if we can use another pedagogical perspectives to further improve the orders. Furthermore, while we did not consider a personalized input of the courses in this work, some models can actually personalize the orders, which is also one of major challenges in our future work.

CHAPTER 5

Conclusion and Future Work

This thesis tackled the problems of course content analysis and course ordering from a knowledge-based perspective. We addressed three tasks and obtained the achievements as follows:

Task 1 *Course content modeling*. We assumed that mapping course content with a domain knowledge categorization helps identify course content in the process of selecting courses. We proposed a method to estimate the relatedness of a course and a domain knowledge category by computing their connections under the Wikipedia article and category structure. The experimental results show that we can estimate the knowledge distribution of a course over the knowledge category *KU* at an accuracy rate of 0.537 in terms of cosine similarity. For the more challenging task, i.e., estimating the relative knowledge distribution, the Wikipedia structure-based method achieves a better performance than the baseline method.

Task 2 *Knowledge coverage estimation*. In this task, we firstly defined the knowledge coverage of a knowledge category by a course as the extent to which the knowledge required in the category is also taught in the course. Then, we proposed a centrality-based computation method to estimate the concept importance to the knowledge categories, which is then aggregated to compute the knowledge coverage. The experimental results show that our method can generate closer knowledge coverage values to the ground truth assigned by human experts, compared to the uniform computation method.

Task 3 *Course ordering*. We addressed the problem of ordering the related courses for a given technical terminology while following the prerequisite relationships among courses. We also evaluated the effectiveness of the orders from pedagogical perspectives such as specific/general fundamentality, and local reference. From the experimental results, we concluded that our proposed method works well in prioritizing related courses and ordering them with a shorter reference distance.

To solve a wider spectrum of problems, we plan to address the followings in future work:

- **Comparison of intrinsic and extrinsic aspects of courses.** In this thesis, we put emphasis on the intrinsic aspect of course content, namely, the knowledge taught in a course. As a clean representation of the knowledge, we utilized the course syllabi in the experiments. However, other course materials such as lecture videos, slides, and assignments demonstrate not only the knowledge but also how it is conveyed in the course. Comparing the body of the knowledge and the way it is presented can provide insights on the difficulty and teaching styles of the courses.

- **Detection of knowledge duplication.** Generally, there is no need to repeat learning the same knowledge if the students prioritize efficiency over steadiness. To this end, identifying the duplicate knowledge is essential when combining multiple courses. Integrating and improving existing works on course segmentation and course content clustering will be a promising direction.

- **Generalization of course ordering problem.** In **Task 3**, we explored the effectiveness of the orders from pedagogical perspectives. It would be more useful if we can generalize the model in which different preferences over the pedagogical perspectives are acceptable. Besides, we did not consider the students' background knowledge and the mastery level of those knowledge. In future work, we plan to personalize the ordering results to various learning goals and learning abilities of students.

- **Applications in other problems.** Although the proposed methods are tailored to solve the educational problems we have defined, they are not limited to solve these problems. For instance, the Wikipedia structure-based method we propose in **Task 1** is supposed to work well in differentiating documents especially when they are semantically close and can be modeled as sets of Wikipedia concepts. In addition, the centrality-based importance estimation in **Task 2** can effectively discriminate a set of objects if a high-quality taxonomy is available. Validating the methods in solving other problems could be an interesting research direction.

Acknowledgements

It has been a long journey for me to take this thesis to completion. I would like to show my gratitude to the people who have supported, guided, encouraged, and inspired me throughout the journey.

My deepest gratitude goes to my supervisor, Professor Masatoshi Yoshikawa. Not only did he open the door of “Computer Science” for me, but he also convincingly guided me to become a computer scientist with his patience, enthusiasm, and expertise. Professor Yoshikawa was always open to my ideas though sometimes they were still immature. He stayed positive and acute to provide new directions when I got frustrated by the unexpected experimental results. Without his continuous support, I would not have been able to overcome the difficulties in undertaking this research.

My sincere appreciation also goes to my co-supervisors, Professor Yasuhito Asano and Associate Professor Kazunari Sugiyama. I could not imagine having learnt more from the frequent discussions and repeated revisions of writing with them. Their harsh questions and generous advice have pushed me to conduct research more precisely and professionally.

I would like to acknowledge the support and encouragement from other academia advisors, Professors Rakesh Agrawal, Toru Ishida, Adam Jatowt, Hajime Kita, Hiroaki Ogata, and Keishi Tajima. I could not be luckier to have the opportunities receiving advice and comments from them, as their expertise in different domains and diverse perspectives have extended my research in both width and depth. Special regards to Professors Ogata and Tajima for being members of the thesis committee and making this thesis closer to its best.

I would like to recognize the invaluable assistance from members of Yoshikawa Ma Laboratory. Associate Professor Qiang Ma, Assistant Profressors Toshiyuki Shimizu and Yang Cao provided me constructive comments and advice on research in every possible way. Ms. Rika Ikebe and Ms. Yuko Nakahara helped me much on academia and personal business, making my research life smooth and joyful. I also wish to thank my fellow labmates, Ms. Ling Xu, Doctors Chenyi Zhuang, Purnomo Husnul Khotimah, Ms. Sora Lim, Doctors Hideaki Ohashi, Wiradee Imrattanatrai, Suppanut Pothirattanachaikul, Mr. Junjie Sun, and Mr. Yang Zhang for all the advice, inspiration, company, and the shared feeling when things got tough.

Last but not the least, I am indebted to my beloved parents. They have always been supporting me to chase my dream, though sometimes it is difficult for them to understand the work I am doing. To keep me concentrated on my research, they had sacrificed a lot and gone through difficult times with my absence. There is no better way for me to pay back but keeping on challenging as a researcher.

Yiling Dai, February 2021

REFERENCES

[1] Andreas M. Kaplan and Michael Haenlein. Higher Education and the Digital Revolution: About MOOCs, SPOCs, Social Media, and the Cookie Monster. *Business Horizons*, 59(4):441–450, 2016.

[2] Yvonne Belanger and Jessica Thornton. Bioelectricity: A Quantitative Approach Duke University's First MOOC, 2013.

[3] Heather B. Shapiro, Clara H. Lee, Noelle E. Wyman Roth, Kun Li, Mine Çetinkaya Rundel, and Dorian A. Canelas. Understanding the Massive Open Online Course (MOOC) Student Experience: An Examination of Attitudes, Motivations, and Barriers. *Computers & Education*, 110:35–50, 2017.

[4] Kai S. Koong, Lai C. Liu, and Xia Liu. A Study of the Demand for Information Technology Professionals in Selected Internet Job Portals. *Journal of Information Systems Education*, 13(1):21–28, 2002.

[5] Timothy C. Lethbridge. What Knowledge Is Important to a Software Professional? *Computer*, 33(5):44–50, 2000.

[6] Cheryl Aasheim, Jordan Shropshire, Lixin Li, and Christopher Kadlec. Knowledge and Skill Requirements for Entry-Level IT Workers: A Longitudinal Study. *Journal of Information Systems Education*, 23(2):193–204, 2012.

[7] Marisa Exter. Comparing Educational Experiences and On-the-Job Needs of Educational Software Designers. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE'14)*, pages 355–360, 2014.

[8] Marisa Exter, Secil Caskurlu, and Todd Fernandez. Comparing Computing Professionals' Perceptions of Importance of Skills and Knowledge on the Job and Coverage in Undergraduate Experiences. *ACM Transactions on Computing Education*, 18(4), 2018.

[9] Yaojie Li, Xuan Wang, and Daqi Xin. An Inquiry into AI University Curriculum and Market Demand: Facts, Fits, and Future Trends. In *Proceedings of the 2019 on Computers and People Research Conference (SIGMIS-CPR'19)*, pages 139–142, 2019.

[10] Ana Sánchez, César Domínguez, Jose Miguel Blanco, and Arturo Jaime. Incorporating Computing Professionals' Know-How: Differences between Assessment by Students, Academics, and Professional Experts. *ACM Transactions on Computing Education*, 19(3), 2019.

[11] Amanpreet Kapoor and Christina Gardner-McCune. Understanding CS Undergraduate Students' Professional Identity through the Lens of Their Professional Development. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'19)*, pages 9–15, 2019.

[12] Guanliang Chen, Dan Davis, Markus Krause, Efthimia Aivaloglou, Claudia Hauff, and Geert-Jan Houben. From Learners to Earners: Enabling MOOC Learners to Apply Their Skills and Earn Money in an Online Market Place. *IEEE Transactions on Learning Technologies*, 11(2):264–274, 2018.

[13] Darco Jansen, Robert Schuwer, Antonio Teixeira, and Cengiz Hakan Aydin. Comparing MOOC Adoption Strategies in Europe: Results from the HOME Project Survey. *The International Review of Research in Open and Distributed Learning*, 16(6), 2015.

[14] Raniah Samir Adham and Karsten Oster Lundqvist. MOOCS As A Method Of Distance Education In The Arab World - A Review Paper. *European Journal of Open, Distance and E-Learning*, 18(1):123–138, 2015.

[15] Aakarsh Shrivastava, Nitasha Hasteer, and K. M. Soni. Perspectives on Curriculum Inclusive MOOCs for Engineering Education: Reflection & Learning. In *Proceedings of 2019 IEEE Tenth International Conference on Technology for Education (T4E)*, pages 34–37, 2019.

[16] Ahmed Tlili, Ronghuai Huang, Ting-Wen Chang, Fabio Nascimbeni, and Daniel Burgos. Open Educational Resources and Practices in China: A Systematic Literature Review. *Sustainability*, 11(18), 2019.

[17] Albert T. Corbett and John R. Anderson. Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1994.

[18] Jose Gonzalez-Brenes, Yun Huang, and Peter Brusilovsky. General Features in Knowledge Tracing to Model Multiple Subskills, Temporal Item Response Theory, and Expert Knowledge. In *Proceedings of the 7th International Conference on Educational Data Mining (EDM2014)*, pages 84–91, 2014.

[19] Siqian Zhao, Chunpai Wang, and Shaghayegh Sahebi. Modeling Knowleddge Acquisition from Multiple Learning Resource Types. In *Proceedings of the 13th International Conference on Educational Data Mining (EDM2020)*, pages 313–324, 2020.

[20] Jie Xu, Tianwei Xing, and Mihaela van der Schaar. Personalized Course Sequence Recommendations. *IEEE Transactions on Signal Processing*, 64(20):5340–5352, 2016.

[21] Asmaa Elbadrawy and George Karypis. Domain-Aware Grade Prediction and Top-n Course Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys'16)*, pages 183–190, 2016.

[22] Rajiv Srivastava, Girish Keshav Palshikar, Saheb Chaurasia, and Arati Dixit. What's Next? A Recommendation System for Industrial Training. *Data Science and Engineering (DSE)*, 3(3):232–247, 2018.

[23] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun. Hierarchical Reinforcement Learning for Course Recommendation in MOOCs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):435–442, 2019.

[24] Zachary A. Pardos and Weijie Jiang. Designing for Serendipity in a University Course Recommendation System. In *Proceedings of the Tenth International Conference on Learning Analytics and Knowledge (LAK'20)*, pages 350–359, 2020.

[25] Chao Wang, Hengshu Zhu, Chen Zhu, Xi Zhang, Enhong Chen, and Hui Xiong. Personalized Employee Training Course Recommendation with Career Development Awareness. In *Proceedings of The Web Conference 2020 (WWW'20)*, pages 1648–1659, 2020.

[26] John Mark Froiland. The Intrinsic Learning Goals of Elementary School Students, in Their Own Words. *Journal of Humanistic Psychology*, 0(0):1–21, 2018.

[27] Jae Hwa Lee and Aviv Segev. Knowledge Maps for E-Learning. *Computers & Education*, 59(2):353–364, 2012.

[28] Jingyun Wang, Takahiko Mendori, and Juan Xiong. A Language Learning Support System Using Course-Centered Ontology and Its Evaluation. *Computers & Education*, 78:278–293, 2014.

[29] R.P. Jagadeesh Chandra Bose, Om Deshmukh, and B. Ravindra. Discovering Concept Maps from Textual Sources. In *Proceedings of the 8th International Conference on Educational Data Mining*, 2015.

[30] Jifan Yu, Chenyu Wang, Gan Luo, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jie Tang. Course Concept Expansion in MOOCs with External Knowledge and Interactive Game. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4292–4302, 2019.

[31] IEEE Standard for Learning Object Metadata. *IEEE Std 1484.12.1-2002*, pages 1–40, 2002.

[32] Danish Contractor, Kashyap Popat, Shajith Ikbal, Sumit Negi, Bikram Sengupta, and Mukesh K. Mohania. Labeling Educational Content with Academic Learning Standards. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 136–144, 2015.

[33] Nicholas Lennox and Justine Diggens. Knowledge, Skills and Attitudes: Medical Schools' Coverage of an Ideal Curriculum on Intellectual Disability. *Journal of Intellectual & Developmental Disability*, 24(4):341–347, 1999.

[34] Laurie E. Macdonald and Kenneth T. Fougere. Software Piracy: A study of the Extent of Coverage in Introductory MIS Textbooks. *Journal of Information Systems Education*, pages 325–329, 2003.

[35] Kiyoshi Ishihata, Hajime Ohiwa, Hiroyasu Kakuda, Kentaro Shimizu, Tetsuo Tamai, Hitoshi Nagasaki, Hidenori Nakazato, Takako Nakatani, Teruo Hikita, Takao Miura, Tatsuo Minohara, Koichi Wada, and Osamu Watanabe. Investigation on the Educational Contents among Informational Science and Engineering Departments by Using Syllabus (Intermediate Report), 2010.

[36] Kornraphop Kawintiranon, Peerapon Vateekul, Atiwong Suchato, and Proadpran Punyabukkana. Understanding Knowledge Areas in Curriculum through Text Mining from Course Materials. In *2016 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 161–168, 2016.

[37] Aditya G. Parameswaran and Hector Garcia-Molina. Recommendations with Prerequisites. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys'09)*, pages 353–356, 2009.

[38] Haiping Zhu, Feng Tian, Ke Wu, Nazaraf Shah, Yan Chen, Yifu Ni, Xinhui Zhang, Kuo-Ming Chao, and Qinghua Zheng. A Multi-Constraint Learning Path Recommendation Algorithm Based on Knowledge Map. *Knowledge-Based Systems (KBS)*, 143:102–114, 2018.

[39] Daqian Shi, Ting Wang, Hao Xing, and Hao Xu. A Learning Path Recommendation Model Based on a Multidimensional Knowledge Graph Framework for E-Learning. *Knowledge-Based Systems (KBS)*, 195:105618, 2020.

[40] Tawanna R. Dillahunt, Brian Zengguang Wang, and Stephanie Teasley. Democratizing Higher Education: Exploring MOOC Use among Those Who Cannot Afford a Formal Education. *The International Review of Research in Open and Distributed Learning*, 15(5), 2014.

[41] Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, New York, NY, USA, 2013. 999133.

[42] Shian-Shyong Tseng, Pei-Chi Sue, Jun-Ming Su, Jui-Feng Weng, and Wen-Nung Tsai. A New Approach for Constructing the Concept Map. *Computers & Education*, 49(3):691–707, 2007.

[43] Marián Šimko and Mária Bieliková. Automated Educational Course Metadata Generation Based on Semantics Discovery. In *Proceedings of the 4th European Conference on Technology Enhanced Learning: Learning in the Synergy of Multiple Disciplines*, pages 99–105, 2009.

[44] Shyi-Ming Chen and Po-Jui Sue. Constructing Concept Maps for Adaptive Learning Systems Based on Data Mining Techniques. *Expert Systems with Applications*, 40(7):2746–2755, 2013.

[45] Takayuki Sekiya, Yoshitatsu Matsuda, and Kazunori Yamaguchi. Curriculum Analysis of CS Departments Based on CS2013 by Simplified, Supervised LDA. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge (LAK'15)*, pages 330–339, 2015.

[46] Evgeniy Gabrilovich and Shaul Markovitch. Computing Semantic Relatedness Using Wikipedia-Based Explicit Semantic Analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07)*, pages 1606–1611, 2007.

[47] Xiaohua Hu, Xiaodan Zhang, Caimei Lu, E. K. Park, and Xiaohua Zhou. Exploiting Wikipedia as External Knowledge for Document Clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pages 389–396, 2009.

[48] Yegin Genc, Yasuaki Sakamoto, and Jeffrey V. Nickerson. Discovering Context: Classifying Tweets Through a Semantic Transform Based on Wikipedia. In *Proceedings of the 6th International Conference on Foundations of Augmented Cognition: Directing the Future of Adaptive Systems (FAC'11)*, pages 484–492, 2011.

[49] Michael Strube and Simone Paolo Ponzetto. WikiRelate! Computing Semantic Relatedness Using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2 (AAAI'06)*, pages 1419–1424, 2006.

[50] Xinpeng Zhang, Yasuhito Asano, and Masatoshi Yoshikawa. A Generalized Flow-Based Method for Analysis of Implicit Relationships on Wikipedia. *IEEE Transactions on Knowledge and Data Engineering*, 25(2):246–259, 2013.

[51] Alberto J. Cañas and Joseph D. Novak. Concept Mapping Using CmapTools to Enhance Meaningful Learning. In *Knowledge Cartography: Software Tools and Mapping Techniques*, pages 25–46. 2008.

[52] Rakesh Agrawal, Behzad Golshan, and Evangelos Papalexakis. Toward Data-Driven Design of Educational Courses: A Feasibility Study. *Journal of Educational Data Mining (JEDM)*, 8(1):1–21, 9 2016.

[53] Yiming Yang, Hanxiao Liu, Jaime Carbonell, and Wanli Ma. Concept Graph Learning from Educational Data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM'15)*, pages 159–168, 2015.

[54] Kotaro Nakayama, Masahiro Ito, Maike Erdmann, Masumi Shirakawa, Tomoyuki Michishita, Takahiro Hara, and Shojiro Nishio. Wikipedia Mining: A Survey on Wikipedia Researches. *Transactions of the Japanese Society for Artificial Intelligence*, 2(4):49–60, 2009.

[55] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. Labeled LDA: A Supervised Topic Model for Credit Attribution in Multi-Labeled Corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 248–256, 2009.

[56] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. Improving Efficiency and Accuracy in Multilingual Entity Extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-SEMANTICS'13)*, pages 121–124, 2013.

[57] Gerard Salton and Christopher Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[58] Yiling Dai, Masatoshi Yoshikawa, and Yasuhito Asano. Estimating Knowledge Category Coverage by Courses Based on Centrality in Taxonomy. *IEICE Transactions on Information and Systems*, E103.D(5):928–938, 2020.

[59] Craig E. Bain, Alan I. Blankley, and L. Murphy Smith. An Examination of Topical Coverage for the First Accounting Information Systems Course. *Journal of Information Systems*, 16(2):143–164, 2002.

[60] Paul Denny, Andrew Luxton-Reilly, John Hamer, and Helen Purchase. Coverage of Course Topics in a Student Generated MCQ Repository. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE'09)*, pages 11–15, 2009.

[61] Dekang Lin. An Information-Theoretic Definition of Similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, pages 296–304, 1998.

[62] Michael Schuhmacher and Simone Paolo Ponzetto. Knowledge-Based Graph Document Modeling. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM'14)*, pages 543–552, 2014.

[63] Christian Paul, Achim Rettinger, Aditya Mogadala, Craig A. Knoblock, and Pedro Szekely. Efficient Graph-Based Document Similarity. In *The Semantic Web. Latest Advances and New Domains(ESWC 2016)*, pages 334–349, May 2016.

[64] Yuan Ni, Qiong Kai Xu, Feng Cao, Yosi Mass, Dafna Sheinwald, Hui Jia Zhu, and Shao Sheng Cao. Semantic Documents Relatedness Using Concept Graph Representation. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM'16)*, pages 635–644, 2016.

[65] Günes Erkan and Dragomir R. Radev. LexRank: Graph-Based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research (JAIR)*, 22(1):457–479, 2004.

[66] Xiaojun Wan and Jianguo Xiao. Exploiting Neighborhood Knowledge for Single Document Summarization and Keyphrase Extraction. *ACM Transactions on Information Systems (TOIS)*, 28(2):8:1–8:34, 2010.

[67] Daraksha Parveen and Michael Strube. Integrating Importance, Non-Redundancy and Coherence in Graph-Based Extractive Summarization. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*, pages 1298–1304, 2015.

[68] Haniyeh Rashidghalam, Mina Taherkhani, and Fariborz Mahmoudi. Text Summarization Using Concept Graph and BabelNet Knowledge Base. In *2016 Artificial Intelligence and Robotics (IRANOPEN)*, pages 115–119, 2016.

[69] Zhuli Xie. Centrality Measures in Text Mining: Prediction of Noun Phrases That Appear in Abstracts. In *Proceedings of the ACL Student Research Workshop (ACLstudent'05)*, pages 103–108, 2005.

[70] Kino Coursey and Rada Mihalcea. Topic Identification Using Wikipedia Graph Centrality. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers (NAACL-Short '09)*, pages 117–120, 2009.

[71] François Rousseau and Michalis Vazirgiannis. Graph-of-Word and TW-IDF: New Approach to Ad Hoc IR. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM'13)*, pages 59–68, 2013.

[72] Linton C Freeman. Centrality in Social Networks Conceptual Clarification. *Social Networks*, 1(3):215–239, 1978.

[73] Gabor Csardi and Tamas Nepusz. The Igraph Software Package for Complex Network Research. *InterJournal of Complex Systems*, 1695(5):1–9, 2006.

[74] M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2):167–256, 2003.

[75] Carter Butts. Social Network Analysis with Sna. *Journal of Statistical Software*, 24(6):1–51, 2008.

[76] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 1994.

[77] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw Hill, New York, 2000.

[78] Paolo Ferragina and Ugo Scaiella. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10)*, pages 1625–1628, 2010.

[79] Janez Brank, Gregor Leban, and Marko Grobelnik. Annotating Documents with Relevant Wikipedia Concepts. In *Proceedings of the Slovenian Conference on Data Mining and Data Warehouses (SikDD 2017)*, 2017.

[80] C.J. Date. *An Introduction to Database Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 8 edition, 2003.

[81] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*. Pearson Education, Inc., New Jersey, 2nd edition, 2008.

[82] Karl Pearson and Olaus Magnus Friedrich Erdmann Henrici. VII. Mathematical Contributions to the Theory of Evolution.-III. Regression, Heredity, and Panmixia. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 187:253–318, 1896.

[83] Shiqiang Guo, Folami Alamudun, and Tracy Hammond. RésuMatcher: A Personalized Résumé-Job Matching System. *Expert Systems with Applications*, 60:169–182, 2016.

[84] Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. Enhancing Person-Job Fit for Talent Recruitment: An Ability-Aware Neural Network Approach. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'18)*, pages 25–34, 2018.

[85] Anik Jacobsen and Gerasimos Spanakis. It's a Match! Reciprocal Recommender System for Graduating Students and Jobs. In *Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019)*, pages 580–583, 2019.

[86] Ahood Almaleh, Muhammad Ahtisham Aslam, Kawther Saeedi, and Naif Radi Aljohani. Align My Curriculum: A Framework to Bridge the Gap between Acquired University Curriculum and Required Market Skills. *Sustainability*, 11(9), 2019.

[87] Chen Liang, Zhaohui Wu, Wenyi Huang, and C. Lee Giles. Measuring Prerequisite Relations Among Concepts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1668–1674, 2015.

[88] Jonathan Gordon, Linhong Zhu, Aram Galstyan, Prem Natarajan, and Gully Burns. Modeling Concept Dependencies in a Scientific Corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, volume 2, pages 866–875, 2016.

[89] Liangming Pan, Chengjiang Li, Juanzi Li, and Jie Tang. Prerequisite Relation Learning for Concepts in MOOCs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, volume 1, pages 1447–1456, 2017.

[90] Mohsen Sayyadiharikandeh, Jonathan Gordon, Jose-Luis Ambite, and Kristina Lerman. Finding Prerequisite Relations using the Wikipedia Clickstream. In *Companion Proceedings of The 2019 World Wide Web Conference (WWW'19 Companion)*, pages 1240–1247, 5 2019.

[91] Fred Paas, Alexander Renkl, and John Sweller. Cognitive Load Theory and Instructional Design: Recent Developments. *Educational Psychologist*, 38(1):1–4, 2003.

[92] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[93] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 4 1994.

[94] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report SIDL-WP-1999-0120, Stanford Digital Library Technologies Project, 1999.

[95] Glen Jeh and Jennifer Widom. Scaling Personalized Web Search. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*, pages 271–279, 2003.

[96] Joseph L Fleiss. Measuring Nominal Scale Agreement among Many Raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[97] A. P. Dawid and A. M. Skene. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28, 1979.

[98] Kalervo Järvelin and Jaana Kekäläinen. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 10 2002.

Selected List of Publications

- **Journals**

[1] Yiling Dai, Masatoshi Yoshikawa and Yasuhito Asano. Estimating Knowledge Category Coverage by Courses Based on Centrality in Taxonomy. *IEICE Transactions on Information and Systems*, E103-D(5):928–938, May 2020.

[2] Yiling Dai, Masatoshi Yoshikawa and Kazunari Sugiyama. Prerequisite-Aware Course Ordering towards Getting Relevant Job Opportunities. *Expert Systems with Applications*. (submitted)

- **International Conferences**

[3] Yiling Dai ,Yasuhito Asano and Masatoshi Yoshikawa. Course Content Analysis: An Initiative Step toward Learning Object Recommendation Systems for MOOC Learners. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM)*, Raleigh, NC, June, 2016.

- **Domestic Conferences and Workshops**

[4] Yiling Dai ,Yasuhito Asano and Masatoshi Yoshikawa. Matching Course Syllabus with Domain Knowledge Space: A Wikipedia Structure-Based Method. In *The 9th Forum on Data Engineering and Information Management (DEIM)*, 2017.

Appendix

A Course information in Chapter 4

Table A.1: Course information used in our experiments.

ID	Course Title	Prerequisite ID
c_1	Computer Programming 1	-
c_2	Computer Programming 2	c_1
c_3	Introduction to Computer Systems	c_2, c_{18}
c_4	Data Structures and Algorithm Analysis	c_2, c_{24}
c_5	Mobile App Development	c_2
c_6	Web Site Design and Development	c_1
c_7	Software Architecture & Design	c_2
c_8	Computer Network Security	c_9
c_9	Computer Networks	c_3, c_4
c_{10}	Operating Systems	c_3, c_4
c_{11}	Human Computer Interaction Design	c_6
c_{12}	Advanced Web Design and Programming	c_4, c_6
c_{13}	Database Systems	c_4
c_{14}	Applied Artificial Intelligence	c_4
c_{15}	Software Engineering	c_7
c_{16}	Computing Science Project	c_{15}
c_{17}	Programming Methods	c_2
c_{18}	Discrete Structure 1 for Computer Science	-
c_{19}	Algorithm Design and Analysis	c_4
c_{20}	Distributed Systems	c_9, c_{10}, c_{13}
c_{21}	Web-based Information Systems	c_{12}, c_{13}
c_{22}	Expert Systems	c_{14}
c_{23}	Systems Software Design	c_{15}
c_{24}	Discrete Structure 2 for Computer Science	c_{18}

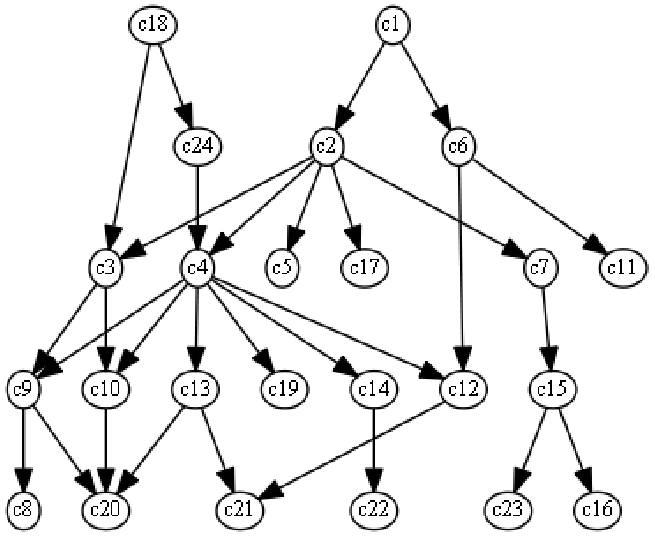


Figure A.1: Course graph information used in our experiments.

B Technical terminology information in Chapter 4

Table B.2: Technical terminologies used in our experiments.

ID	Technical terminology	ID	Technical terminology
t_1	Web application	t_{35}	Computer program
t_2	Object-oriented programming	t_{36}	Web service
t_3	Database	t_{37}	World Wide Web
t_4	SQL	t_{38}	Computer hardware
t_5	Knowledge representation and reasoning	t_{39}	Machine learning
t_6	Programming language	t_{40}	Subroutine
t_7	JavaScript	t_{41}	Graphical user interface
t_8	Software development process	t_{42}	Software bug
t_9	Software testing	t_{43}	Knowledge base
t_{10}	HTML	t_{44}	Unit testing
t_{11}	Operating system	t_{45}	Debugging
t_{12}	Microsoft SQL Server	t_{46}	Data management
t_{13}	Java virtual machine	t_{47}	Agile software development
t_{14}	Software engineering	t_{48}	Modular programming
t_{15}	Java (programming language)	t_{49}	JavaServer Pages
t_{16}	Cascading Style Sheets	t_{50}	Java Platform, Enterprise Edition
t_{17}	Software maintenance	t_{51}	Data structure
t_{18}	MySQL	t_{52}	Test case
t_{19}	PHP	t_{53}	Stored procedure
t_{20}	Formal specification	t_{54}	Algorithmic efficiency
t_{21}	Active Server Pages	t_{55}	Transact-SQL
t_{22}	Web server	t_{56}	Hypertext Transfer Protocol
t_{23}	Computer network	t_{57}	Relational database
t_{24}	Scripting language	t_{58}	HTML5
t_{25}	Unix	t_{59}	Test automation
t_{26}	ASP.NET	t_{60}	Client-server model
t_{27}	Ajax (programming)	t_{61}	Software deployment
t_{28}	Computational problem	t_{62}	Internet protocol suite
t_{29}	Software design pattern	t_{63}	Software project management
t_{30}	Algorithm	t_{64}	Software documentation
t_{31}	Digital signature	t_{65}	Apache HTTP Server
t_{32}	User interface	t_{66}	Version control
t_{33}	Web development	t_{67}	Java servlet
t_{34}	JQuery		