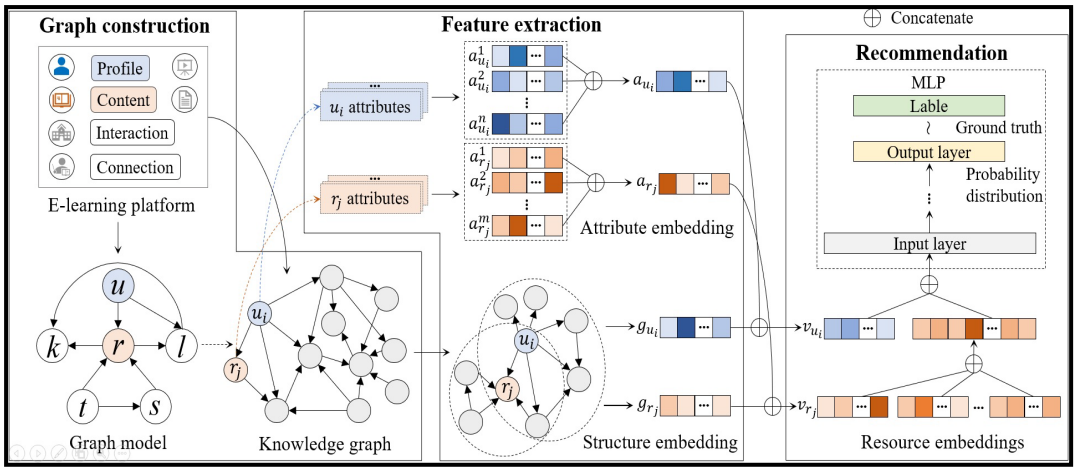
Resource recommendation in e-learning platforms based on knowledge graph

Qing Tang

Par **Qing TANG**

Resource recommendation in e-learning platforms based on knowledge graph

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 18 décembre 2023
Spécialité : Informatique : Unité de recherche Heudyasic (UMR-7253)
D2786

Université de technologie de Compiègne

Thèse de Doctorat

Resource recommendation in e-learning platforms based on knowledge graph

Par **Qing TANG**

Directrices de thèse
Marie-Hélène Abel & Elsa Negre

Spécialité : Informatique

Soutenue le 18 décembre 2023

Jury :

Dominique LENNE	Université de technologie de Compiègne	Président
Nada MATTA	Université de technologie de Troyes	Rapporteur
Philippe ROOSE	Université de Pau et des Pays de l'Adour	Rapporteur
Armelle BRUN	Université de Lorraine	Examinateur
Djamal BENSLIMANE	Université Claude Bernard Lyon 1	Examinateur
Marie-Hélène ABEL	Université de technologie de Compiègne	Directrice
Elsa NEGRE	Université Paris Dauphine	Directrice

Thèse soumise pour l'obtention du diplôme de Doctorat en informatique

CID (Connaissances, Incertitudes, Données)

Laboratoire Heudiasyc

UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

Abstract

CID (Connaissances, Incertitudes, Données)
Heudiasyc

Doctorat en informatique

Resource recommendation in e-learning platforms based on knowledge graph

by Qing TANG

As an integral part of education, e-learning provides users with massive resources and allows them learn independently without the constraints of time and space. However, flooded with such a huge number of resources offered by different individuals and institutions, personalized learning is required to help users choose resources precisely. Recommender systems are widely used in e-learning platforms to reduce the information burden and achieve personalized learning. Generally speaking, the more information a recommender system knows, the more accurate the recommendations will be. But conventional recommendation approaches usually ignore a pivotal piece of information, the latent connections between entities in the e-learning platform. Collaborative filtering approaches make recommendations by learning user-resource historical interactions (e.g., user ratings and browsing history), content-based approaches mainly consider the relevant information of users and resources (e.g., user profile and resource content). In addition to user and resource, there are numerous entities lurking in e-learning platform, and the latent connections between these entities can provide crucial information support to the function of recommender system. For example, users may choose resources created by authors with whom they are familiar, i.e., there are latent connections between users and authors, and the connections affect users' choices.

Graph based technologies (e.g., knowledge graph) have the capacity to extract, represent, manipulate, and model information (including latent connections) in different domains. In this context, we propose a recommendation approach incorporating knowledge graph to recommend pedagogical resources to e-learning platform users. We integrate knowledge graph, feature extraction and neural network into a recommendation framework. It contains three modules: *i)* modeling the information of e-learning platform via knowledge graph; *ii)* acquiring features of users and resources from the formed knowledge graph; *iii)* learning the acquired features with neural network for resource recommendation.

To evaluate the performance of the recommendation framework, we conduct a series of experiments based on two datasets from real-world e-learning platforms, and the results confirm that the proposed recommendation framework outperforms the methods from the literature.

Résumé

En tant que partie intégrante de l'éducation, l'e-learning fournit aux utilisateurs des ressources considérables et leur permet d'apprendre de manière autonome sans les contraintes de temps et d'espace. Cependant, face au grand nombre de ressources proposées par différents individus et institutions, une aide personnalisée au choix de ces dernières semble nécessaire pour faciliter l'apprentissage. Les systèmes de recommandation sont largement utilisés dans les plateformes e-learning pour réduire la surcharge d'information et parvenir à apprentissage personnalisé. En général, plus un système d'information connait d'informations plus les recommandations seront pertinentes. Mais les approches de recommandation conventionnelles ignorent généralement les informations issues des relations que peuvent entretenir les entités utilisées par les plateformes e-learning. Les approches de filtrage collaboratif font des recommandations en apprenant les interactions historiques utilisateur-ressource (par exemple, les évaluations des utilisateurs et l'historique de navigation), les approches basées sur le contenu prennent principalement en compte les informations pertinentes sur les utilisateurs et les ressources (par exemple, le profil utilisateur et le contenu des ressources). En plus de l'utilisateur et de la ressource, de nombreuses entités se cachent dans la plateforme e-learning, et les relations que peuvent entretenir ces entités et le support informationnel qu'elles peuvent offrir à un système de recommandation. Par exemple, un utilisateur peut préférer choisir des ressources créées par un auteur spécifique, c'est-à-dire qu'il existe une connexion latente entre l'utilisateur et l'auteur, et cette connexion affecte les choix de l'utilisateur.

Les technologies basées sur les graphes (par exemple, le graphe de connaissance) ont la capacité d'extraire, de représenter, de manipuler et de modéliser des informations dans différents domaines. Dans ce contexte, nous proposons une approche de recommandation intégrant graphe de connaissance pour recommander des ressources pédagogiques aux utilisateurs de plateformes e-learning. Nous intégrons le graphe de connaissance, l'extraction de fonctionnalités et le réseau de neurones dans un cadre de recommandation. Un tel graphe comporte trois modules: *i)* modélisation des informations de la plateforme e-learning via graphe de connaissances; *ii)* acquérir des fonctionnalités d'utilisateurs et de ressources à partir du graphe de connaissances formé; *iii)* apprendre les fonctionnalités acquises avec réseau de neurones pour la recommandation de ressources.

Pour évaluer les performances du cadre de recommandation, nous conduisons une série d'expériences basées sur deux ensembles de données provenant de plateformes e-learning du monde réel, et les résultats confirment que le cadre de recommandation proposé surpasse les méthodes de la littérature.

Acknowledgements

From 2019 to 2023, I spent a memorable time at laboratory Heudiasyc of Université de Technologie de Compiègne (UTC) in order to obtain my Ph.D degree. At this moment, I want to sincerely thank all those who made this a success.

I would like to express my sincere gratitude to my supervisors Mme Marie-Hélène Abel and Mme Elsa Negre. They have guided me in the scientific research with their profound experience. And great thanks also should be given to the jury members Mme Armelle Brun, Mme Nada Matta, M. Dominique Lenne, M. Djamal Benslimane, and M. Philippe Roose for their efforts and insightful reviews that refined and advanced my research.

Besides, I would like to thank my colleagues from Heudiasyc and other departments of UTC. They have given me a lot of help and support in my work.

Moreover, I want to thank all my friends in Compiègne. All the wonderful times I experienced with you will become my precious memories.

Last but not least, I sincerely appreciate the financial support from China Scholarship Council (CSC) and Heudiasyc, which enables me to carry out my work in good conditions.

Contents

List of Tables

List of Figures

List of Symbols

$u, u\prime$	user
$i, i\prime$	item
c	course
t	teacher
G	knowledge graph
$\langle h, r, t \rangle$	triplet of knowledge graph
e_h	head entity embedding of triplet
e_r	relation embedding of triplet
e_t	tail entity embedding of triplet
d	dimension
$\mathfrak{E}$	set of entities
$\mathfrak{R}$	set of relations
$r_{u,i}$	ground truth rating between user u and item i
$\hat{r}_{u,i}$	predicted rating between user u and item i
U	set of users
I	set of items
$\overline{r}_u, \overline{r}_{u\prime}$	average value of the ratings provided by all users for i and $i\prime$
σ_u	variance value of the ratings provided by all users for item i
$\mathcal{R}, \mathcal{P}, \mathcal{Q}$	rating matrix in SVD
q_i, p_i	user vector and item vector in SVD
b_u, b_i	biases associated with u and i in SVD
S_test	subset of the ratings used to evaluate the RS
$L_{rec}(u)$	item list recommended to u
$rel_{u,i}$	relevance value between u and i
v_u	vector representation of user u
v_i	vector representation of item i
v_{a_u}	attribute feature vector of user u
v_{a_r}	attribute feature vector of resource r
v_{g_u}	structure feature vector of user u
v_{g_r}	structure feature vector of resource r

List of Abbreviations

RS	**Recommender System**
CF	**Collaborative Filtering**
CB	**Content Based**
KG	**Knowledge Graph**
RDF	**Resource Description Framework**
NLP	**Natural Language Processing**
KGE	**Knowledge Graph Embedding**
MLP	**MultiLayer Perceptron**
MOOC	**Massive Open Online Course**
LMS	**Learning Management System**
CMS	**Course Management System**
LSS	**Learning Support System**
PCC	**Pearson Correlation Coefficient**
SVD	**Singular Value Decomposition**
NCF	**Neural Collaborative Filtering**
TF	**Term Frequency**
IDF	**Inverse Document Frequency**
KNN	**K Nearest Neigbors**
SVM	**Support Vector Machine**
RMSE	**Root Mean Squared Error**
MAE	**Mean Absolute Error**
MAP	**Mean Average Precision**
NDCG	**Normalized Discounted Cumulative Gain**
HR	**Hit Rate Machine**
GNN	**Graph Neural Network**

GCN	**Graph Convolutional Network**
GAN	**Graph Attention Network**

Part I

INTRODUCTION

Chapter 1

Introduction

1.1 Research Context

With the rapid development of computer science, the application of advanced technologies (e.g., machine learning and deep learning) have brought a considerable impact on our daily lives. Rapidly advancing computer technologies have made remote tools increasingly popular, which is also the case in education. We have witnessed the development and transformation of learning in the past decades, including the changes of channels for knowledge transmission and environments for teaching and learning [Li et al., 2022].

E-learning is defined as the learning supported by digital tools and information media, emerging based on the integration of information and communication technologies in education field [Tayebinik et al., 2013]. E-learning has been using and developing for many years [Downes, 2005] [Horton, 2011] [Martin et al., 2020] due to the significant advantages such as easy access to massive resources [Aczel et al., 2008], flexibility in time and space [Behera, 2013], cost-effective strategy [Frehywot et al., 2013]. In the past few years, Covid-19 has greatly promoted the development of e-leaning, and even during the military war in Ukraine, its importance has been demonstrated [Matviichuk et al., 2022]. The war destroyed educational facilities and deprived the opportunities of traditional learning, resulting in the massive requirements for e-learning.

E-learning platform is an important application of e-learning in education industry. A variety of e-learning platforms have been designed over the past period, with typical commercial e-learning platforms such as Blackboard and Desire2Learn, and open-source e-learning platforms such as ILIAS, Moodle, and Sakai [Piotrowski, 2010]. Another representative group of e-learning platforms is the products of Massive Open Online Courses (MOOCs), such as Coursera, edX, and Udacity, offering thousands of users accesses to countless courses from institutions around the world [Gong et al., 2020]. However, e-learning platforms usually suffer from information overload and users often struggle in determining the quality and suitability of resources, especially when they do not have enough knowledge to critically review them.

Recommender system (RS) is one of the most prominent technologies for accelerating the pace of e-learning [Ezaldeen et al., 2022], it helps users select appropriate resources to reduce the information burden caused by massive resources of e-learning platforms and thereby improves their learning efficiency. RSs are generally categorized into three groups: Collaborative Filtering (CF) RS, Content Based (CB) RS, and hybrid RS [Zhang et al., 2019b]. Pure CF RS ignores the content information about users and items, and focuses only on the feedback from users (mainly provided in the form of ratings assigned to items). It may suffer from data sparsity (rating matrix contains a large number of null values) and cold start problem (ratings are not enough to make reliable predictions). CB RS makes recommendations by comparing the representations of item contents with the representations of user interests, and focuses on modeling users and items (mainly through building models for them). It may suffer from over specialization (suggesting items directly related to the user profiles rather than new items). Hybrid RS combines two or more recommendation techniques and uses different types of data (e.g., ratings, item content, and user profile) with the aim of improving the performance of CF RS or CB RS and mitigating their flaws.

However, in e-learning platforms, conventional recommendation approaches usually ignore a piece of crucial information, the latent connections. In addition to user and resource, there are numerous entities lurking in e-learning platform, and the latent connections between these entities can provide crucial information support to the function of RS. For example, in an e-learning platform, a user enrolls a course not because the content of the course is more appropriate for the user, but because the user prefers the teaching style of the teacher who teaches the course. It means that, the latent connections between user, course and teacher influence the user's choice. Figure 1.1 gives an example to illustrate the limitation of conventional CF recommendation approach.

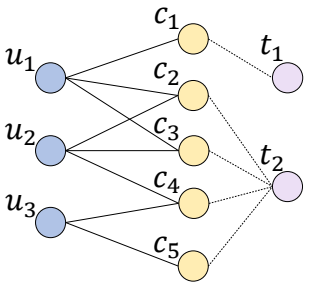
User	Course
u_1	c_1, c_2, c_3
u_2	c_2, c_3, c_4
u_3	c_4, c_5

Teacher	Course
t_1	c_1
t_2	c_2, c_3, c_4, c_5

(a) Information of dataset

	c_1	c_2	c_3	c_4	c_5
u_1	1	1	1		
u_2		1	1	1	
u_3				1	1

(b) Rating matrix



(c) Knowledge graph

Figure 1.1: Limitation of conventional CF recommendation approach.

Figure 1.1(a) shows the interactions between users (u), courses (c), and teachers (t) in an e-learning platform. With the conventional CF RS, the rating matrix can be built, as shown in Figure 1.1(b), thereby user u_1 and user u_2 will be deemed more similar. So course c_1 that enrolled by u_1 but not by u_2 will be recommended to u_2. But what if u_2 prefers the courses taught by teacher t_2? If we characterize the data in terms of KG, as shown in Figure 1.1(c), there is a possibility that both u_2 and u_3 prefer the courses taught by t_2. If so, c_5 should be recommended to u_2, not c_1.

Knowledge representation is an important branch of artificial intelligence, it has the capacity to extract, represent, manipulate, and model information in different application domains [Ji et al., 2021]. It provides standardized tools, such as Knowledge Graph (KG), that allowing machines to efficiently manage information and capture latent information. As a representative tool of knowledge representation, KG has powerful ability of characterization, it can be built to model facts in the form of Resource Description Framework (RDF) triplets by connecting the entities (i.e., nodes) through relations (i.e., edges).

1.2 CHALLENGES AND RESEARCH QUESTIONS

Building RS for e-learning platform is a very complex process that involves many procedures such as data collection, data preprocessing, feature extraction, and model fitting. The general recommendation framework is shown as Figure 1.2.

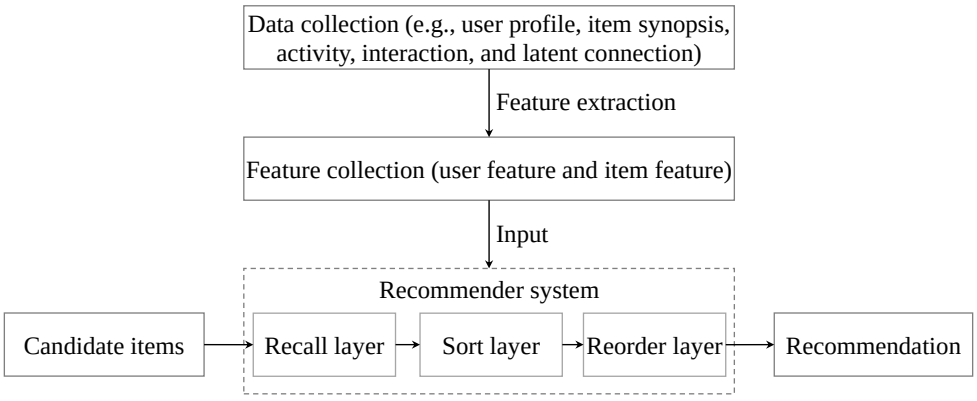


Figure 1.2: General recommendation framework.

As the input to RS, the quality of feature collection is closely related to the performance of RS. In the process of feature extraction, features can be extracted from ordinary data, such as user profiles, item contents, and user-item interactions (i.e., ratings); at the same time, latent features may be extracted from the latent connections between the entities of e-leaning

platform. How to capture these latent connections and make it machine readable? Our interest centers on the choice of using KG to characterize the content of e-learning platform in order to capture the latent connections and extract the latent features.

In simple terms, we propose to use KG to obtain more valuable information that cannot be obtained by conventional approaches, enhancing the feature extraction of recommendation framework (see Figure 1.2), thereby improving the performance of downstream RS. So, the first research question of this thesis is:

Research Question 1. How to characterize the content of e-learning platform by the means of KG? Can the KG be generic?

Generally, RS makes recommendations by learning the features of users and items (e.g., similarity and model-based analysis). In our case, after characterizing the e-learning platform by the means of KG, the next step is to obtain features of users and resources (item is called resource in e-learning platforms) from the result (i.e., formed KG). However, a formed KG not only contains the semantic descriptions and values of the resources and users (i.e., ordinary data), but also the latent connections as Figure 1.1 illustrated. So, the second research question of this thesis is:

Research Question 2. How to obtain features of users and resources from the constructed KG? How to learn these features with algorithm for recommendation?

1.3 MAIN CONTRIBUTION

To answer the **Research Question 1**. We follow the 4 core steps to construct a lightweight ontology represented as a KG to characterize the content of an e-learning platform [Noy et al., 2001]. *step 1.* Determine the domain and scope of the ontology; *step 2.* Define the entity classes and their relation classes; *step 3.* Define the classes of attributes associated with entity classes; *step 4.* Extract the instances of the defined classes and compress them into a knowledge base represented as a KG. Particularly, the classes can be adjusted according to different e-learning platforms, thereby making it generic.

To answer the **Research Question 2**. The formed KG of an e-learning platform contains the semantic descriptions and values of the resources and users, as well as the latent connections between multiple entities. First, we use Natural Language Processing (NLP) to process the semantic descriptions and values of the resources and users to get text feature vectors of users and resources. Then, we use Knowledge Graph Embedding (KGE) to learn the latent connections of the formed KG to get structure

feature vectors of users and resources. Finally, the obtained text and structure feature vectors are concatenated to form the overall feature vectors of users and resources, respectively, and input into a Multilayer Perceptron (MLP) network for model fitting. The trained MLP model has capability to predict the relevance between different users and resources and make resource recommendations to target users.

In summary, we propose a recommendation framework based on KG to recommend pedagogical resources to e-learning platform users. The framework consists of three modules: *graph construction* characterizes the content of e-learning platform via KG; *feature extraction* acquires features of users and resources from the formed KG; *recommendation* learns the acquired features with MLP for resource recommendation.

To evaluate the performance of the recommendation framework, we conducted a series of experiments by using datasets form real-world e-learning platforms. The results demonstrate that our recommendation framework outperforms methods from the state of the art on the datasets.

The python code using open-source software library PyTorch is available at https://github.com/qingtang3009/MOOC-Recommendation.

1.4 DISSERTATION ORGANIZATION

The rest of this thesis is organized as follows:

Part II State of the art introduces the related technologies and previous studies regarding our research questions. It consists of three chapters:

> **Chapter 2 E-learning.** This chapter introduces e-learning, e-learning platforms, and user models in e-learning platforms.
>
> **Chapter 3 Knowledge graph.** This chapter introduces KG, most representative KG models, and KGE algorithms.
>
> **Chapter 4 Recommender system.** This chapter first introduces the context of RS. Then, it presents the classification of RSs, CF RS, CB RS, and hybrid RS. Next, it presents the applications of KG in RS. Finally, it presents the applications of RS in e-learning.

Part III Contributions elaborate the proposed recommendation framework, and show the procedures and results of the experiments conducted using the real-world datasets. It consists of two chapters:

> **Chapter 5 Recommendation framework based on knowledge graph.** This chapter elaborates the details of the proposed recommendation framework, including graph construction, feature extraction (text embedding and structure embedding), and recommendation.
>
> **Chapter 6 Experiments.** This chapter conducts a series of experiments to evaluate the performance of our proposed recommendation framework and compares it with the state-of-the-art methodologies.

Part IV Conclusions and perspectives conclude this thesis and make perspectives for the future work. It contains one chapter:

> **Chapter 7 Conclusions and perspectives.** This chapter summarizes the proposed framework, points out where improvements can be made as well as the possible approaches, and looks ahead to the future work.

Part II

STATE OF THE ART

Chapter 2

E-learning

2.1 Introduction

E-learning is a branch of education and has been developing for many years [Downes, 2005] [Horton, 2011] [Tayebinik et al., 2013] [Martin et al., 2020]. The advancement of computer technologies has made remote tools more and more popular, which is also the case in education field. The term e-learning has emerged under the combination of information and communication technologies [Tayebinik et al., 2013].

Some of the definitions of e-learning are given below:

- E-learning covers a wide set of processes and applications that users can perform online activities to teach themselves with the help of multimedia materials, such as Web, radio, video, and TV [Hassenburg, 2009].
- E-learning refers to the usage of computer based technologies, mainly through the Internet, to deliver knowledge and instructions to individuals [Wang et al., 2010].
- E-learning can be viewed as computer assisted pedagogy for user-centered and collaborative learning, it comprises all forms of electronically supported learning and teaching [Jethro et al., 2012].
- E-learning is a general term for different ways of transmitting knowledge, such as Web based learning, virtual classroom, and digital collaboration. It includes the delivery of information via Internet, intranet, satellite broadcast, interactive TV, and CD-ROM [Norén Creutz et al., 2014].

Based on the above definitions, we can summarily define e-learning as:

Definition 1 *E-learning represents a class of learning methods that utilize a wide range of tools based on information and communication technologies for supporting the transfer of knowledge.*

Under the influence of external factors, e-learning shows adaptive development trend, especially during the Covid-19 pandemic in the past few

years. According to the research, Covid-19 has a positive impact on the popularization of e-learning and the innovation of e-learning tools in the college education [Stecuła et al., 2022b], the percentage of users who are familiar with the e-learning tools has significantly increased during the pandemic. There has been a visible rise for users to study through e-learning platforms, especially the following platforms: MS Teams, Zoom, and Google classroom [Stecuła et al., 2022a].

On the other hand, the telecommunications and the broadband sectors have increased accessibility to economical Internet connectivity plans, which also provides the possibility for more and more people to enjoy the convenience brought by Internet technologies. According to the survey of Global Market Insight[1], nearly 4.9 billion individuals use the Internet, while it was 4.1 billion in 2019. As the number of Internet users increases, more and more people will be able to access e-learning platforms to take courses or complete degrees. It is worth mentioning that the younger generations have been raised with the usage of mobile technologies and the Internet, they follow Web channels, play computer games, record and post videos which makes them more receptive to emerging Web technologies. Thus, e-learning is a great opportunity to better adapt to the expectations of younger generations [Rajeh et al., 2021]. To put it simply, technologies bring the chances of using e-learning, remote education meets the needs of young generations, and educators should strive to apply the emerging technologies to teaching purposes, thereby promoting the development of education.

E-learning can be divided into synchronous learning and asynchronous learning. In synchronous learning, teachers and users join together and learn in a real-time situation, the learning process is collaborative and facilitated on the virtual platform, such as live streaming, video conference, and live chatting [Fernandez et al., 2022]. The benefit of synchronous learning is that users can get immediate feedback and question-answer can be conducted effectively in real-time as the sessions proceeding. In asynchronous learning, there is a time gap between teachers and users, and many representative methods are used in asynchronous learning, such as prerecorded session, virtual library, social media platform, and online forum [Lin et al., 2012]. The benefit of asynchronous learning is that it offers a lot of flexibility for users to progress in their own learning rhythms without limitation of schedule from teachers.

In summary, e-learning plays a significant role in the filed of education for any country, it offers great opportunities to enhance their development of education which could benefit all mankind. As the main approach to realize e-learning, e-learning platform has attracted more and more attention. A variety of standardized and user-oriented e-learning platforms have been lunched, and we will introduce typical ones in the next section.

[1] https://www.gminsights.com/

2.2 E-LEARNING PLATFORMS

E-learning relies on technologies, i.e., the realization of e-learning requires hardware, software, and network infrastructure. Theoretically, any software or service that provides a learning environment for its users can be called e-learning platform, such as e-mail, chatting software, forum, blog, and collaboration tool. These different kinds of support can be used individually or in various combinations as e-learning platform. But this kind of setup has obvious drawbacks, such as the lack of common user management, the limited interoperability between the tools, and the difficulty for resource authentication [Piotrowski, 2010]. Thus, the development trend of e-learning platform has been moving towards to more equitable, standardized, and universal.

Nowadays, numerous standardized e-learning platforms have been created and they are available for general users with equitable opportunities. Typical examples such as Moodle and MOOCs, which have existed for many years before the Covid-19 pandemic. Their usage has been enforced by the influence of pandemic as institutions try to overcome barriers to maintaining regular educational efforts. And there are many alternative names for e-learning platforms, such as Learning Management System (LMS), Course Management System (CMS), Virtual Learning Environment (VLE), Managed Learning Environment (MLE), and Learning Support System (LSS) [Piotrowski, 2010]. These platforms may differ in many ways but they share one characteristic: supporting and realizing the e-learning [Bichsel, 2013]. They allow institutions to deliver learning materials to their users with 24/7 access and free location.

Information storage is an essential technology in e-learning platforms, usually realized by a standard relational database management system [Piotrowski, 2010]. Some typical examples are MySQL or PostgreSQL for open-source systems and Oracle or Microsoft SQL Server for commercial systems. As for the underlying logic, e-learning platforms are based on a variety of programming languages and system architectures. Some platforms are written in PHP (e.g., ILIAS and Moodle), some platforms are written in Java (e.g., OLAT and Sakai), and some others are based on two languages (e.g., Blackboard is written in a mix of Perl and Java). Many platforms are open-source, they are usually developed by communities and free to register (but may charge in site).

We present two typical examples, Moodle and MOOCs.

Moodle

Moodle is the abbreviation of modular object-oriented dynamic learning environment, which is a free and open-source LMS, written in PHP and distributed under the GNU General Public License[2]. The Moodle 1.0

[2] https://www.gnu.org/

was released in 2002 with the combination of PHP, Apache Web server, and MySQL. Nowadays, Moodle is actually an e-learning platform that serves as an important tool to achieve blended learning [Horvat et al., 2015]. It allows the information exchange among users geographically dispersed, through synchronous (e.g., chatting) or asynchronous (e.g., forum) channels. Users can create custom websites with online courses and join community-sourced plugins on Moodle [Porter, 2013]. From a functional perspective, Moodle has easily configurable features, allowing users to create assessment processes (e.g., quiz, test, and survey) and manage tasks with their timetable, as well as offering a wide variety of complementary tools to support the teaching or learning process.

The functionalities of Moodle can be grouped into two classes: resources and modules [Blin et al., 2008]. Resources represent instructional materials that are usually created in digital formats and then uploaded to the platform, such as Web pages, slideshows, word documents, flash animations, video and audio files. Modules are components created via Moodle in order to achieve interactions among students and teachers towards manipulation and content transformation, such as databases, lessons, assignments, workshops, chats, forums, news, quizzes, surveys, feedback, shareable contents and some external tools [Costa et al., 2012]. Most of the activities on Moodle take place inside courses (i.e., few functions can be used outside the context of a specific course) and the activities can be classified into 6 classes: creation, organization, delivery, communication, collaboration, and assessment [Piotrowski, 2010].

MOOCs

MOOCs is the abbreviation of massive open online courses, to be more precise, it represents a series of open online learning platforms. The application of MOOCs is one of the most prominent trends in higher education in recent years. The term 'MOOCs' represents open access, video-based online courses, and massive resources released through online platforms for supporting the learning of users [Baturay, 2015]. It first appeared in 2008 based on connective-distributed peer learning model. After that, a few educational videos were created by the professors from Stanford University and released through open online platforms supported with free web resources in 2011, and this was the year that MOOCs exploded around the world. In the past few years, the number of users and courses in the platforms of MOOCs has been continue growing, it holds millions of registered users and thousands of lunched courses around the world. Most of the MOOCs users enroll courses on the basis of professional objectives, and interest users also account for a certain proportion.

Nowadays, the number and diversity of MOOCs continue to grow and gain an increased popularity among both students and educators in higher education [Aldowah et al., 2020]. MOOCs are often released by third

party online platforms and developed independently by academics [Baturay, 2015]. Over 900 universities around the world had launched 11,400 MOOCs until 2018 [Aldowah et al., 2020]. Many scholars and practitioners consider MOOCs as a way to enhance equity in higher education because of their potential to reach a wider audience and to remove barriers to high-quality education offered by elite institutions [Deng et al., 2019].

Basically, the course in MOOCs has multiple elements that need to function properly. Table 2.1 presents the main elements and corresponding descriptions [Grainger, 2013].

Table 2.1: Main elements of the course in MOOCs.

Element	**Description**
Video lecture	video lectures are the main products of MOOCs, they are usually prerecorded and can be reused in certain scenarios;
Assessment	assessments are used to measure users' knowledge levels, mainly through the auto-graded quizzes;
Forum	forums are places where users post questions and other users or teachers reply;
Reading	readings are the electronic versions of books in traditional classrooms and most of them are available online or provided by teachers;
Live session	live sessions exist as a supplement to weekly video lectures;
Activity	instructional activities are offered with the aim of allowing users to further test their understanding of the knowledge concepts;
Scripted video	scripted videos are used to enhance users' comprehension of scenes;
Social media	users are encouraged to continue their learning-related discussions on other social media platforms.

There are three main reasons why people choose MOOCs: *1)* users want gain more knowledge in the fields or topics they are interested in; *2)* users want to get certificates or degrees while applying for jobs; *3)* some teachers guide their students to MOOCs to reinforce, support or prepare them for their in-class learning [Baturay, 2015].

Despite the obvious advantages of MOOCs over the traditional education, many different challenges are still found, and one of the most notable challenge is the dropout [Kim et al., 2017]. Several reports show that the completion rate in the platforms of MOOCs is very low compared to the number of users enrolled in these courses and therefore a high dropout rate exists [Feng et al., 2019]. Some researchers state that most dropouts

occur in the early stages of learning, which needs further exploration [Breslow et al., 2013] [Said, 2017] [Chen et al., 2019]. Similarly, Coursera's Social Network Analysis reported that only 2% of participants have completed the courses [Aldowah et al., 2020]. Since MOOCs are becoming more and more popular all over the world, researchers and educational developers are starting to explore innovative ways to help users participating in these courses to persist longer and learn more [Barak et al., 2016] [Hadi et al., 2016] [Chen et al., 2019]. In summary, more and more researchers have devoted themselves to the study of MOOCs and its platforms, with the aim of improving the learning efficiency and learning experience of users.

2.3 USER MODEL IN E-LEARNING PLATFORM

The information about users is essential for any kind of e-learning platform. Without user information, platforms are not able to adapt themselves to users' characteristics and preferences. The required information can be preserved and managed in the form of user model.

Before introducing user model for information preserving in e-learning platforms, it is necessary to clearly distinguish data, information, and knowledge, because they are easily confused.

Data is 'raw' recorded symbols; information is data processed or transformed into a form or structure suitable for being used by human (i.e., information is the refinement of data for the purposes of human usage); knowledge is what someone has after understanding information [Kendal et al., 2007]. Note that, data, information, and knowledge are not static things, but stages in the process of using data and transforming it into knowledge. The stages from data to knowledge implies a shift from facts to more abstract concepts, as shown in Figure 2.1.

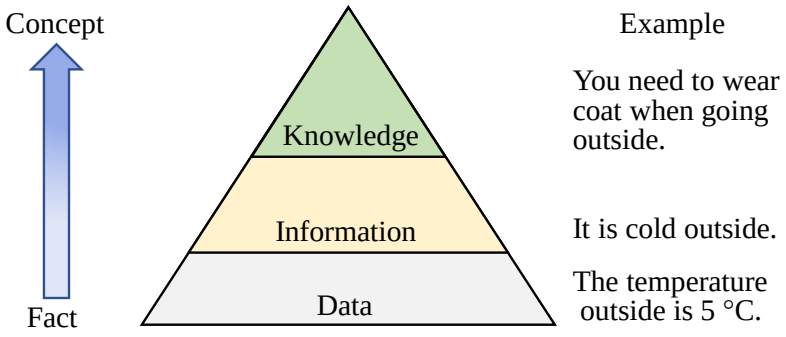


Figure 2.1: Data, information and knowledge [Kendal et al., 2007].

User model mainly contains processed data about the user, i.e., user model contains information or knowledge about the user, which is mainly obtained by analyzing the data of the user.

The terms user profile and user model are often used as synonyms or only one term is used by meaning of both. Here, we first clearly distinguish them.

User profile is a collection of raw user data, which is preserved without adding further description, interpretation, or inference; user model is a specific representation of the user, which should contain the needed characteristics of the user regarding the context of application [Froschl, 2005]. Depending on the content and the amount of data, user profile can be modeled. In other words, user model can be derived by refining user profile.

The importance of user model is reflected in the fact that it should be constructed whenever a personalized request is expected and many applications can benefit from it, such as information support, search assistant, and recommendation. Furthermore, not only the positive or neutral information of a user needs to be extracted and preserved to construct a user model, but also the limitations of the user (e.g., color blindness and disability) [Froschl, 2005].

The information of user model can be divided into two major groups: domain-independent information and domain-specific information.

Domain-independent information

The domain-independent information mainly refers to the user information without any direction or domain. Having this information available, either directly or indirectly, may help e-learning platform to get a sufficient information support to enhance the user's learning experience. It usually includes background, preference, motivation, and basic skill.

- *Background.* Background information is created when user registers on e-learning platform, including name, age, gender, e-mail, highest education, profession, etc.
- *Preference.* Users have inherent preferences related to some aspects when choose resources, such as language preference and format preference. These preferences are considered to be separated from the learning process.
- *Motivation.* The motivations answer the questions why users utilize e-learning platforms and what learning goals they want to achieve.
- *Basic skill.* Users have generic skills that are not limited to any specific domain, such as English reading level, comprehension ability, and reasoning ability.

Domain-specific information

The domain-specific information represents a reflection of the user's knowledge level or competency in a particular domain, such as JAVA software development capability and autopilot algorithm knowledge level. Accurately grasping the domain-specific information of users is a key factor to provide personalized services to users.

There are three common channels for obtaining a user's domain-specific information: *1)* prior tests can be used to determine the prior knowledge level of users before they start learning in specific domains; *2)* users' activity records can be used to analyze their knowledge levels; *3)* the results of assessments and evaluations generated by users during the learning processes are direct reflections of their current knowledge levels in the specific domains [Tang et al., 2022].

2.4 CHAPTER SUMMARY

In this chapter, we began with a detailed introduction of e-learning, including its definitions, development, strengths, and weaknesses. Then, we introduced e-learning platforms and presented two representative examples, Moodle and MOOCs. Finally, we introduced user model, which can provide information support about user for downstream tasks (e.g., recommendation) in e-learning platforms.

To conclude, the popularization and development of e-learning can not be separated from the assistance of e-learning platform, more and more resources have been placing in varies e-learning platforms, which brings great convenience to users' learning. But on the other hand, the excessive amount of resources also creates an information burden for users. They often struggle in determining the suitability of resources, especially when they do not have enough knowledge to critically review the resources. We focus on helping users filter appropriate learning resources to alleviate the information burden and improve their learning efficiency in e-learning platforms.

Chapter 3

Knowledge graph

3.1 Introduction

As a representative technology of knowledge representation, Knowledge Graph (KG) has continued receiving attention from both industry and academia since its inception. KG can effectively organize and represent knowledge so that it can be utilized in advanced applications, such as Web searching and knowledge reasoning [Chen et al., 2020]. Due to the increasing amount of Web resources and release of linked open data projects, many KGs have been constructed [Chen et al., 2020], and they are also needed in e-learning platforms [Ilkou, 2022].

Before detailing the related technologies of KG, we clarify its definition. Some of the definitions are given below:

- KG uses graph-based model to capture knowledge in application scenarios that involve integrating, managing and extracting data from diverse sources at large-scale [Hogan et al., 2021].
- KG defines possible classes of entities and relations, organizes real world data in a graph, allowing arbitrary entities to be potentially interconnected and cover a variety of subject areas [Cimiano et al., 2017].
- KG is a graph consists of a set of RDF triplets, where each triplet $\langle \mathfrak{h}, \mathfrak{r}, \mathfrak{t} \rangle$ is an ordered set of the following RDF terms: a subject $\mathfrak{h}$, a predicate $\mathfrak{r}$, and an object $\mathfrak{t}$ [Färber et al., 2018].

Based on the above definitions, we can summarily define KG as:

Definition 2 *KG is a data structure for representing facts related to a specific domain, whose nodes represent entities of the domain and edges represent relations between them.*

The vigorous development of KG began after the announcement of Google in 2012, followed by further usage of KG by enterprises such as Facebook, IBM, LinkedIn, Microsoft, and Uber [Noy et al., 2019] [Hogan et al., 2021]. The growing industrial usage of KG has provoked considerable influence in academia, more and more scientific researches on KG

have been published in recent years [Pujara et al., 2013] [Wang et al., 2014] [Wang et al., 2014] [Ehrlinger et al., 2016] [Wang et al., 2017] [Paulheim, 2017]. KGs can be categorized into open KG or enterprise KG. Open KGs are published online, making their content accessible for the public, such as BabelNet [Navigli et al., 2012], DBpedia [Lehmann et al., 2015], Freebase [Bollacker et al., 2007], Wikidata [Vrandečić et al., 2014], and YAGO [Hoffart et al., 2011], covering many domains and either extracted from sources such as Wikipedia [Hoffart et al., 2011] [Navigli et al., 2012] [Lehmann et al., 2015] or built by volunteers of communities [Bollacker et al., 2007] [Vrandečić et al., 2014]. Enterprise KGs are typically internal to companies and applied for commercial usage. The major industry applications that use enterprise KGs include Web search, information extraction, personal agency, business analysis, risk assessment, automation, advertising, and recommendation [Hogan et al., 2021].

KG construction is a prerequisite for using KG in advanced applications. There are several approaches to construct KGs, one typical approach is to pre-define a conceptual graph model (similar to T-box of ontology) and import instances into a graph based on the model [Zhou et al., 2022]. Yahoo states that they construct their KG by aligning new entities, relations and information with their common ontology [Blanco et al., 2013]. [Ehrlinger et al., 2016] express that KG is somehow superior to ontology with larger scale (e.g., KG is a large ontology) and additional functions (e.g., a built-in reasoner that allows new knowledge to be derived).

3.2 KNOWLEDGE GRAPH MODELS

We introduce three KG models most commonly used in practice.

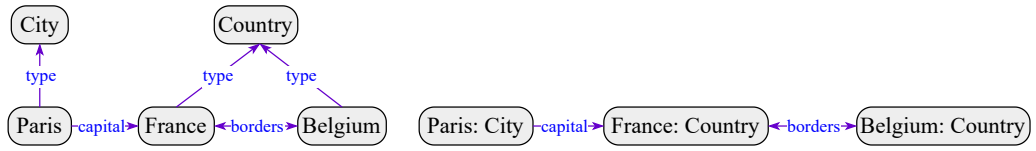
Multi-relational graph

A multi-relational graph is defined as a set of nodes, such as *Paris*, *Montmartre*, *Sacré-Cœur*, and a set of directed labeled edges between those nodes, such as *Sacré-Cœur* $\xrightarrow{\text{in}}$ *Montmartre* and *Montmartre* $\xrightarrow{\text{in}}$ *Paris*. In multi-relational KG, nodes represent entities (e.g., the city Paris, the location Montmartre, and the building Sacré-Cœur) and edges represent binary relations between those entities (e.g., the building Sacré-Cœur is located at Montmartre and the location Montmartre is in the city Paris).

Heterogeneous graph

A heterogeneous graph [Hussein et al., 2018] [Wang et al., 2019b] [Yang et al., 2020] is a graph where each node or edge is assigned one type. Heterogeneous graph is similar to multi-relational graph, with edge labels corresponding to edge types, but where the type of node forms part of the graph model itself, rather than being expressed as a special relation

[Hogan et al., 2021]. The difference between multi-relational graph and heterogeneous graph is shown in Figure 3.1. An edge is homogeneous if it is between two nodes with the same type, such as the borders in Figure 3.1(b); otherwise, it is heterogeneous, such as the capital in Figure 3.1(b). However, unlike multi-relational graph, it typically assumes one-to-one relations between nodes and types [Hogan et al., 2021].

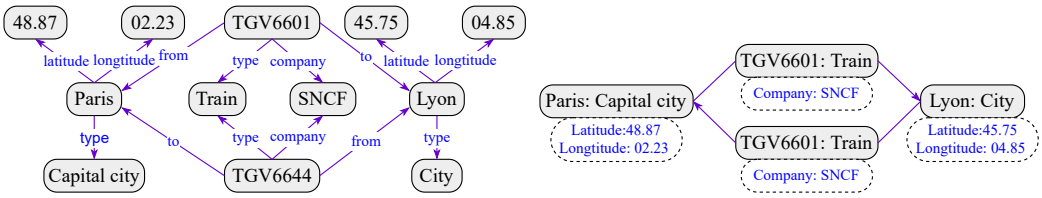


(a) Multi-relational graph (b) Heterogeneous graph

Figure 3.1: Illustrations of (a) multi-relational graph and (b) heterogeneous graph.

Property graph

Property graph allows a set of property–value pairs and labels to be associated with nodes and edges [Miller, 2013] [Angles et al., 2017]. In a multi-relational graph, we cannot directly annotate an edge like $Paris \xrightarrow{\texttt{train}} Lyon$ with the company, but we can add a new node denoting a train and connect it with the source, destination, companies, and type, as shown in Figure 3.2(a).



(a) Multi-relational graph (b) Property graph

Figure 3.2: Illustrations of (a) multi-relational graph and (b) property graph.

Applying this pattern to a large graph may require significant efforts. Differently, Figure 3.2(b) exemplifies a property graph with analogous data, where property–value pairs on edges model companies, property–value pairs on nodes indicate latitudes and longitudes, and node/edge labels indicate the type of node/edge. Though not yet standardized, property graph is used in many popular graph databases, such

as Neo4j [Angles et al., 2017] [Miller, 2013]. Although property graph has more flexibility in terms of how to model data compare to multi-relational graph and heterogeneous graph, it requires more complex intricate query languages, formal semantics, and inductive techniques [Hogan et al., 2021].

3.3 KNOWLEDGE GRAPH EMBEDDING

Machine learning can be directly used for refining a KG [Paulheim, 2017], or downstream tasks with a given KG, such as recommendation [Zhang et al., 2016], information extraction [Vashishth et al., 2018], question answering [Huang et al., 2019], and logical query [Hamilton et al., 2018]. However, machine learning techniques typically assume numeric representations (i.e., vectors), distinguishing it from how graphs are usually expressed. So, how can KG be encoded numerically for machine learning?

Knowledge Graph Embedding (KGE) is a dominating approach for that numerical encode procedure, due to its ability of creating dense vector representations of KG. Before detailing KGE, we clarify its definition:

Definition 3 *Given a KG composed of a collection of triplets* $G = \{\langle \mathfrak{h}, \mathfrak{r}, \mathfrak{t} \rangle\}$, *KGE aims to represent each entity* $\mathfrak{h}, \mathfrak{t} \in \mathfrak{E}$ *and relation* $\mathfrak{r} \in \mathfrak{R}$ *into continuous vector spaces (can be an uniform space or different spaces depending on the algorithms), where* $\mathfrak{E}$ *and* $\mathfrak{R}$ *indicate the sets of entities and relations, respectively.*

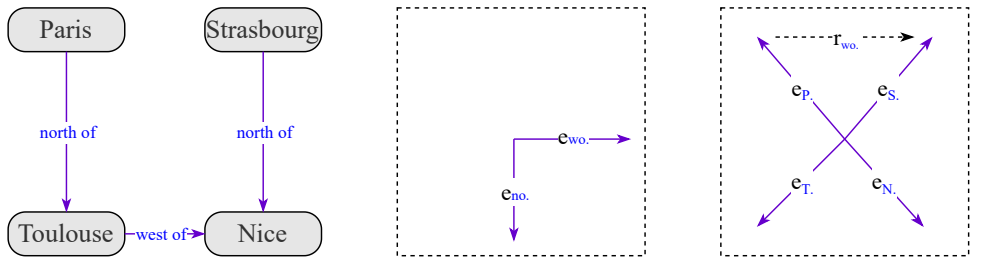
Typically, KGE is composed of an entity embedding for each node (represents entity as vector with $\mathfrak{d}$ dimensions) and a relation embedding for each edge (represents relation as vector with $\mathfrak{d}$ dimensions). The dimension $\mathfrak{d}$ of the embedding is fixed and usually low (e.g., $50 \leqslant \mathfrak{d} \leqslant 1000$). The meaning of these vectors is to abstract and preserve the information of KG. For example, given a triplet $\langle \mathfrak{h}, \mathfrak{r}, \mathfrak{t} \rangle$, the embedding approach defines a score function that accepts $e_{\mathfrak{h}}$ (the entity embedding of head node $\mathfrak{h}$), $e_{\mathfrak{r}}$ (the relation embedding of edge $\mathfrak{r}$), $e_{\mathfrak{t}}$ (the entity embedding of tail node $\mathfrak{t}$), and computes the plausibility (i.e., how likely $\mathfrak{h} \xrightarrow{\mathfrak{r}} \mathfrak{t}$ to be true). Usually, negative sampling (creates negative triplets by randomly replacing the head node or tail node with other entities that do not belong to the entity set of the KG) is employed in KGE. The goal is to maximize the plausibility of positive triplets and minimize the plausibility of negative triplets according to the given score function.

A wide range of KGE algorithms have been proposed [Wang et al., 2017], and they can be summarized into three groups: translation model, tensor factorization model, and neural model.

Translation model

Translation model interprets relation as transformation from head node to tail node [Hogan et al., 2021]. For example, given a triplet

$\langle Marseille, train, Nice \rangle$, translation model sees $train$ as a transformation from $Marseille$ to $Nice$, i.e., $Marseille \xrightarrow{train} Nice$. A seminal translation model is TransE [Bordes et al., 2013]. For a positive triplet, TransE learns the vectors e_h, e_r, e_t, and makes $e_h + e_r$ as close as possible to e_t; on the contrary, it keeps them as far as possible for a negative triplet. Figure 3.3 provides an example of two-dimensional ($\mathfrak{d} = 2$) TransE. In Figure 3.3(a),



(a) Original graph (b) Relation embedding (c) Entity embedding

Figure 3.3: Example of two-dimensional TransE.

we keep the orientation of vectors similar to the original KG for clarity. In this example, $e_{P.}$ represents the embedding of entity $Paris$, $e_{T.}$ represents the embedding of entity $Toulouse$, $e_{wo.}$ represents the embedding of relation $west\ of$, and $e_{N.}$ represents the embedding of entity $Nice$. TransE updates the embeddings while learning from these experiences (e.g., $e_{T.} + e_{wo.} \approx e_{N.}$ and $e_{T.} + e_{wo.} \neq e_{P.}$). These trained embeddings can be used to predict undefined nodes or edges in the KG. For example, they can predict which node in the KG is most likely to be west of $Strasbourg$ by computing $e_{S.} + e_{wo.}$, where $e_{S.}$ represents the embedding of entity $Strasbourg$. The result embedding is closest to $e_{P.}$ (dotted arrow in Figure 3.3(c)), thereby $Paris$ is the most plausible node for this case.

However, TransE is not capable enough in the one-to-many or many-to-many scenarios [Hogan et al., 2021]. For example, taking the train from $Paris$ can not only go to $Lyon$, but also can go to $Marseille$, where TransE tries to give similar embeddings to all target locations. To resolve this problem, many variants of TransE have been investigated, typically using a distinct hyperplane (e.g., TransH [Wang et al., 2014]) or vector space (e.g., TransR [Lin et al., 2015], TransD [Ji et al., 2015]) for each type of relation. Besides, [Sun et al., 2019] propose a translation model, named RotatE, it embeds entities and relations in complex spaces, allowing to capture more characteristics of relations, such as direction, symmetry, inversion, asymmetry, and composition. Other non-Euclidean space embedding methods, such as MuRP [Balazevic et al., 2019] embeds multi-relational graph data in the Poincaré ball model of hyperbolic space, whose curvature provides more space to separate entities with defined dimension.

Tensor factorization model

Tensor factorization decomposes a tensor into lower-order tensors, and these tensors can be seen as capturing latent factors in the original tensor [Hogan et al., 2021]. In such case, a KG can be encoded as a one-hot 3-order tensor, i.e., $G = |X| \times |Y| \times |Z|$, where the position $G_{xyz} = 1$ if the x-th node links to the y-th node with the z-th edge, otherwise $G_{xyz} = 0$. There are many tensor factorization models, and a basic approach computes a KG such that $x_1 \otimes y_1 \otimes z_1 + \cdots + x_d \otimes y_d \otimes z_d \approx G$, as illustrated in Figure 3.4. X, Y, and Z denote the matrices formed by vectors $[x_1 \cdots x_d]$, $[y_1 \cdots y_d]$, and $[z_1 \cdots z_d]$, respectively. Each vector forming a matrix column, one column of Y can be extracted as one relation embedding and the columns of X and Z at the same position can be extracted as embeddings for one entity. However, KGE typically aims to assign one vector to each entity.

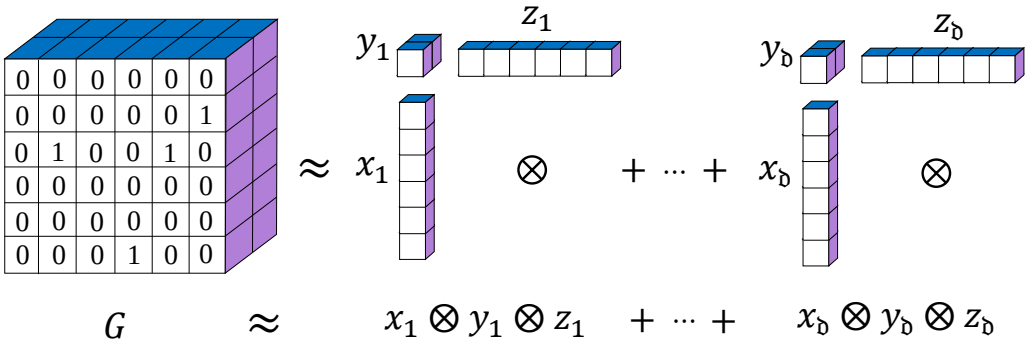


Figure 3.4: Illustration of basic tensor factorization model.

DistMult [Yang et al., 2014] is a seminal method for KGE based on rank factorization, where each entity or relation is associated with a vector of dimension d. For a relation $h \xrightarrow{r} t$, a plausibility score function $\sum_{i=1}^{d}(e_h)_i(e_r)_i(e_t)_i$ is defined, where $(e_h)_i$, $(e_r)_i$, $(e_t)_i$ denote the i-th elements of vectors e_h, e_r, e_t, respectively. Then, the goal is to learn embeddings for each node and edge that maximize the plausibility of positive triplets and minimize the plausibility of negative triplets. This approach equates to the basic model, but the entities have one vector is used twice. A weakness of this approach is that per the score function, the plausibility of $h \xrightarrow{r} t$ will always be equal to $t \xrightarrow{r} h$, i.e., DistMult does not capture edge direction.

Rather than use a vector as a relation embedding, RESCAL [Nickel et al., 2013] uses a matrix, which allows for combining values from e_h and e_t across all dimensions and thus can capture edge direction. However, RESCAL incurs a higher cost in terms of space and time than DistMult. Recently, ComplEx [Trouillon et al., 2016] and HolE [Nickel et al., 2016]

both use vectors for relation and entity embeddings, but ComplEx uses complex vectors, while HolE uses a circular correlation operator [Hogan et al., 2021] to capture edge direction. Other tensor factorization models such as SimplE [Kazemi et al., 2018] and TuckER [Balažević et al., 2019b]. SimplE computes a basic factorization by averaging terms across X, Y, Z to get the final plausibility score. TuckER decomposition outputs a tensor $\mathscr{Z}$ and three three matrices A, B, C, such that $G = \mathscr{Z} \times A \times B \times C$, where entity embeddings are extracted from A and C, and relation embeddings are extracted from B. Among these approaches, TuckER currently provides state-of-the-art results on standard benchmarks.

Neural model

Neural model uses neural networks to obtain graph embeddings with non-linear score function [Hogan et al., 2021]. An early neural model for KGE is Semantic Matching Energy (SME). It learns parameters (i.e., weights: w and $w\prime$) for two functions $f_w(e_h, e_r)$ and $g_{w\prime}(e_t, e_r)$, and the dot product of the results from both functions gives the plausibility score. Later, many variants of f_w and $g_{w\prime}$ are proposed [Bordes et al., 2014]. Another neural model is Neural Tensor Networks (NTN). It maintains a tensor $\mathcal{W}$ of weights and computes plausibility scores by combining the outer product $e_h \otimes \mathcal{W} \otimes e_t$ with e_r and a standard neural layer over e_h and e_t [Socher et al., 2013]. The tensor $\mathcal{W}$ yields a high number of parameters, limiting scalability [Wang et al., 2017]. Multilayer Perceptron (MLP) is a simpler model, where e_h, e_r, and e_t are concatenated and fed into a hidden layer to compute the plausibility score [Dong et al., 2014].

More recent neural models use convolutional kernels. ConvE models the interactions between input entities and relations by convolutional and fully-connected layers [Dettmers et al., 2018]. The main characteristic of ConvE model is that the score is defined by a convolution over 2D shaped embeddings. HypER uses a hypernetwork architecture that generates simplified relation-specific convolutional filters to make the model more intuitive [Balažević et al., 2019a].

3.4 CHAPTER SUMMARY

In this chapter, we introduced KG related technologies, including KG, graph models for KG construction, and KGE for obtaining the feature vectors of KG.

To conclude, the rapid growth of e-learning has led to more and more universities and commercial institutions joining e-learning platforms, various resources are flooding the platforms, and improving user experience has become a top priority for many platforms, such as personalized services. However, an important basis for improving user experience is to obtain enough valuable information from platforms. In e-learning platforms,

in addition to some regular visible data, there is also a lot of invisible information that plays an important role in improving the user experience. KG can effectively organize and represent information, whether visible or invisible, so that it can be utilized in advanced applications. Thus, we propose to use KG to characterize the content of e-learning platform and extract necessary information, KGE is used to obtain the information (features of users and resources) from KG in that process.

Chapter 4

Recommender system

4.1 Introduction

The constantly growing amount of resources in e-learning platforms encourage more and more researchers focus on helping users filter appropriate learning resources to alleviate the information burden and improve their learning efficiency. Recommender System (RS) has been an effective strategy for such a context.

RS is information filtering system designed and applied to find the information most relevant to users' needs and transfer it to users [Negre, 2015]. The utility of RS cannot be underestimated, given it widespread adoptions in many Web applications, along with its effective impact to ameliorate problems related to information overload [Zhang et al., 2019b].

Generally, information from two sources is usually considered in the process of building RSs, user-related information and item-related information. The user-related information mainly refers to user profile and user feedback. The item-related information mainly refers to item content, which can be described by different types of data such as unstructured data (e.g., text, image, and video), semi-structured data (e.g., table with characteristics), or even a mixture of the two. The user profile has been presented in section 2.3, so we focus on user feedback in the following part of this section. User feedback can be divided into two types, explicit feedback and implicit feedback.

- *Explicit feedback* appears as the form of ratings that users assign to items. Ratings are often entered on a defined scale which indicate users' levels of appreciation to items. Typically, ratings are interval-based, where a discrete set of ordered numbers is used to quantify one's likes and dislikes. Many Web applications (e.g., Coursera, Amazon, and Netflix) use a range of values from 1 to 5 as ratings (see Figure 4.1). In this context, a rating of 1 indicates extreme disgust while 5 indicates maximum liking. Formally, in scientific community, the term $r_{u,i}$ is used to represent the rating that user u assigns to item i, the prediction of the rating that user u would assign to item i is denoted as $\hat{r}_{u,i}$.

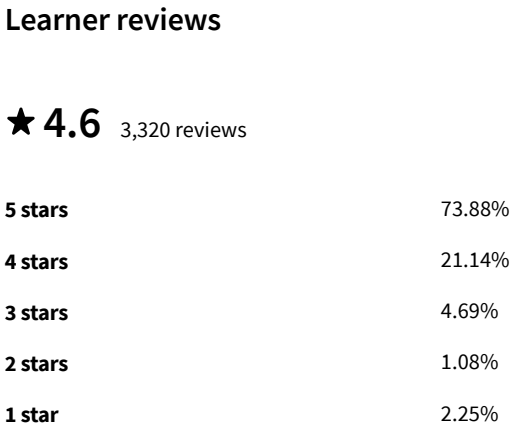


Figure 4.1: Example of explicit feedback on the scale of 1 to 5 on Coursera.

- *Implicit feedback* appears as the form of user-item interactions, allowing them to implicitly show their interests in the items with which they interacted. For example, if a user frequently listens to the music of an artist, it is reasonable to assume that the user likes this artist. Likewise, if a user buys a dress and does not return it, we can assume that the user likes it. Unlike explicit ratings which use ordinal numbers to express user interests, implicit tastes are often represented by interactions.

In a RS, user feedback, whether explicit or implicit, can be represented by a matrix whose rows represent users and columns represent items. Figure 4.2 illustrates an example of two rating matrices, where the rating value assigned by user u for item i is stored in the box located in the u-th row and the i-th column of the matrix.

	i_1	i_2	i_3	i_4	i_5
u_1				2	
u_2	2		4		
u_3					
u_4		5		3	
u_5					1

(a) Explict ratings

	i_1	i_2	i_3	i_4	i_5
u_1				1	
u_2	1		1		
u_3					
u_4		1		1	
u_5					1

(b) Implict ratings

Figure 4.2: Illustrations of (a) explicit feedback and (b) implicit feedback rating matrices. The scale of explicit ratings is 1 to 5 and the value of implicit rating is 0 or 1 ($r_{u,i}$ is 1 if there is an interaction between user u and item i, 0 otherwise).

4.2 CLASSIFICATION OF RECOMMENDER SYSTEMS

RSs are usually classified into three categories: CF RS, CB RS, and hybrid RS [Zhang et al., 2019b].

4.2.1 *Collaborative filtering recommender systems*

CF is widely used in the RSs of both scientific and industrial fields with the objective of predicting the missing rating values. CF RSs make recommendations by learning user-item interactions, either through explicit (e.g., user ratings) or implicit feedback (e.g., browsing history) [Zhang et al., 2019b]. The CF algorithms can be divided into two subcategories that depend on the prediction method: the memory-based CF approach and the model-based CF approach [Breese et al., 2013].

4.2.1.1 *Memory-based collaborative filtering*

The main strategy of memory-based CF approach is based on two assumptions: *1)* similar users rate items in an similar way; *2)* similar items receive similar ratings. It has two types: user-based type and item-based type.

- *User-based type.* Ratings created by users similar to user u are used to make recommendations for u. The rating prediction between user u and item i, $\hat{r}_{u,i}$, is calculated as the weighted average of the ratings for i created by the similar neighbors of u.

- *Item-based type.* Ratings created by u for items similar to item i are used to make recommendations for u. The $\hat{r}_{u,i}$ is predicted based on the specific ratings created by u for these similar neighbors of i.

Whether user-based type or item-based type, memory-based CF consists of two consecutive steps: *i*) identification of the similar neighbors (users or items); *ii*) prediction of ratings based on the identified neighbors. Generally, users ($u \in U$) and items ($i \in I$) are represented by rating vectors in the rating matrix. The vector representations of users $v_u = (r_{u,i}, i = 1, 2, \cdots, |I|)$ and items $v_i = (r_{u,i}, u = 1, 2, \cdots, |U|)$ can be obtained from the matrix.

To identify the similar neighbors, there are some common metrics for calculating similarity between vectors in this context.

Cosine similarity is a metric widely used in the field of information retrieval and text mining to calculate the similarity between two vector representations [Breese et al., 2013]. It calculates the cosine angle between two vectors in d-dimensional space, with d being the length of the vectors. Formally, the cosine similarity between two users (u and u') is defined by the equation 4.1.

$$sim_{cos}(u, u\prime) = cos(\vec{u}, \vec{u\prime}) = \frac{\vec{u} \cdot \vec{u\prime}}{\|\vec{u}\|_2 \times \|\vec{u\prime}\|_2} = \frac{\sum_{i \in I_{u,u\prime}} r_{u,i} r_{u\prime,i}}{\sqrt{\sum_{i \in I_{u,u\prime}} r_{u,i}^2} \sqrt{\sum_{i \in I_{u,u\prime}} r_{u\prime,i}^2}} \quad (4.1)$$

where $I_{u,u\prime}$ represents the set of items rated both by u and $u\prime$. Similarly, the similarity between two items (i and $i\prime$) is defined by the equation 4.2.

$$sim_{cos}(i, i\prime) = cos(\vec{i}, \vec{i\prime}) = \frac{\vec{i} \cdot \vec{i\prime}}{\|\vec{i}\|_2 \times \|\vec{i\prime}\|_2} = \frac{\sum_{i \in U_{i,i\prime}} r_{u,i} r_{u\prime,i}}{\sqrt{\sum_{u \in U_{i,i\prime}} r_{u,i}^2} \sqrt{\sum_{u \in U_{i,i\prime}} r_{u\prime,i}^2}} \quad (4.2)$$

where $U_{i,i\prime}$ represents the set of users who have rated both item i and item $i\prime$.

Another widely used similarity metric in this context is the Pearson Correlation Coefficient (PCC) [Resnick et al., 1994]. PCC measures a correlation between two variables, its value varies between -1 and 1. The value of 1 reflects a perfect and positive correlation while the value of -1 means that the two variables are negatively correlated or anti-correlated. The value of 0 means that there is no linear correlation between the two variables. Formally, the PCC similarity between two users u and $u\prime$ is defined by equation 4.3.

$$sim_{PCC}(u, u\prime) = \frac{\sum_{i \in \mathbf{I}_{u,u\prime}} (r_{u,i} - \bar{r}_u)(r_{u\prime,i} - \bar{r}_{u\prime})}{\sqrt{\sum_{i \in \mathbf{I}_{u,u\prime}} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in \mathbf{I}_{u,u\prime}} (r_{u\prime,i} - \bar{r}_{u\prime})^2}} \quad (4.3)$$

where $\bar{r}_u$ and $\bar{r}_{u\prime}$ represent the average value of the ratings of user u and user $u\prime$, respectively. The PCC similarity between two items i and $i\prime$ is defined by equation 4.4.

$$sim_{PCC}(i, i\prime) = \frac{\sum_{u \in \mathbf{U}_{i,i\prime}} (r_{u,i} - \bar{r}_i)(r_{u\prime,i} - \bar{r}_{i\prime})}{\sqrt{\sum_{i \in \mathbf{U}_{i,i\prime}} (r_{u,i} - \bar{r}_i)^2 \sum_{i \in \mathbf{U}_{i,i\prime}} (r_{u\prime,i} - \bar{r}_{i\prime})^2}} \quad (4.4)$$

where $\bar{r}_i$ and $\bar{r}_{i\prime}$ represent the average value of the ratings provided by all users for item i and item $i\prime$, respectively.

To predict the ratings based on the identified similar neighbors, the weighted average of the ratings created by the individuals in the neighbors of a user need to be calculated in the case of user-based type, or the weighted average of the ratings associated with the elements in the neighbors of an item need to be calculated in the case of item-based type.

Formally, the prediction of $\hat{r}_{u,i}$ is obtained by equation 4.5 for the user-based type.

$$\hat{r}_{u,i} = \frac{\sum_{u\prime \in U\prime_{u,i}} r_{u\prime,i} sim(u,u\prime)}{\sum_{u\prime \in U\prime_{u,i}} sim(u,u\prime)} \tag{4.5}$$

where the term $U\prime_{u,i}$ represents the set of nearest neighbors of the user u who rated the item i. Similarly, in the case where the prediction is based on the items, $\hat{r}_{u,i}$ is obtained by equation 4.6.

$$\hat{r}_{u,i} = \frac{\sum_{i\prime \in I\prime_{u,i}} r_{u,i\prime} sim(i,i\prime)}{\sum_{i\prime \in I\prime_{u,i}} sim(i,i\prime)} \tag{4.6}$$

where the term $I\prime_{u,i}$ represents the set of nearest neighbors of the item i having been rated by u.

However, different rating systems have different standards for rating. In order to better predict, normalization is adopted as an important solution [Herlocker et al., 2002]. A distribution of r_u ratings can be normalized through the mean (average value) $\bar{r}_u$ and its variance σ_u. Two normalization methods are usually considered: mean normalization and z-score normalization (standardization).

The mean normalization for the above two types are shown as equation 4.7 and equation 4.8.

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{u\prime \in U\prime_{u,i}} [(r_{u\prime,i} - \bar{r}_{u\prime}) sim(u,u\prime)]}{\sum_{u\prime \in U\prime_{u,i}} sim(u,u\prime)} \tag{4.7}$$

$$\hat{r}_{u,i} = \bar{r}_i + \frac{\sum_{i\prime \in I\prime_{u,i}} [(r_{u,i\prime} - \bar{r}_{i\prime}) sim(i,i\prime)]}{\sum_{i\prime \in I\prime_{u,i}} sim(i,i\prime)} \tag{4.8}$$

The z-score normalization for the above two types are shown as equation 4.9 and equation 4.10.

$$\hat{r}_{u,i} = \bar{r}_u + \sigma_u \frac{\sum_{u\prime \in U\prime_{u,i}} \left[\frac{(r_{u\prime,i} - \bar{r}_{u\prime})}{\sigma_{u\prime}} sim(u,u\prime)\right]}{\sum_{u\prime \in U\prime_{u,i}} sim(u,u\prime)} \tag{4.9}$$

$$\hat{r}_{u,i} = \bar{r}_i + \sigma_i \frac{\sum_{i\prime \in I\prime_{u,i}} \left[\frac{(r_{u,i\prime} - \bar{r}_{i\prime})}{\sigma_{i\prime}} sim(i,i\prime)\right]}{\sum_{i\prime \in I\prime_{u,i}} sim(i,i\prime)} \tag{4.10}$$

4.2.1.2 *Model-based collaborative filtering*

In contrast to memory-based CF, model-based CF uses the collection of ratings to learn a model (usually a supervised machine learning model), which is then used to make rating predictions [Breese et al., 2013]. Popular models such as Singular Value Decomposition (SVD) and neural network [Aggarwal et al., 2016c].

SVD

The core idea of SVD is to identify latent semantic factors. In the context of CF, SVD is usually used to make the rating matrix dense [Sarwar et al., 2000], and it is one of the most popular matrix factorization models where the rating matrix is considered as a multiplication of two matrices, as shown in equation 4.11.

$$\mathcal{R} \approx \mathcal{P}\mathcal{Q}^T \tag{4.11}$$

where $\mathcal{R}$ is the rating matrix with size $|U| \times |I|$, $\mathcal{P}$ is a matrix with size $|U| \times \mathfrak{d}$ and $\mathcal{Q}$ is a matrix with size $|I| \times \mathfrak{d}$. $\mathfrak{d}$ can be seen as the number of latent factors (i.e., latent dimension). The columns of $\mathcal{P}$ are the latent vectors corresponding to latent factors in the rows of $\mathcal{P}$. Matrix factorization allows users and items to be described in a same latent space of dimension $\mathfrak{d}$, so that the interactions between users and items can be modeled as scalar products in the same space.

Formally, in SVD, each user is represented as vector $p_u \in \mathbb{R}^{\mathfrak{d}}$ and each item is represented as vector $q_i \in \mathbb{R}^{\mathfrak{d}}$. The dot product of the vectors (each user vector and item vector pair) indicates the correlation between user and item. To predict the $\hat{r}_{u,i}$, the result is the dot product of p_u and q_i. SVD consider both the user-item interactions and the personalized features (bias) of users and items, thus, the $\hat{r}_{u,i}$ can be calculated as equation 4.12.

$$\hat{r}_{u,i} = p_u \cdot q_i^T + b_u + b_i \tag{4.12}$$

where b_u and b_i can be interpreted as the biases associated with the user u and item i.

To learn model parameters, the objective function is performed as equation 4.13.

$$L = \arg \min_{p_u, q_i, b_u, b_i} \sum_{r_{u,i} \in R} (r_{u,i} - p_u \cdot q_i^T - b_u - b_i) \tag{4.13}$$

Neural collaborative filtering

In recent years, deep learning has been widely used in the field of CF RS [Zhang et al., 2019b]. Because deep learning makes it possible to effectively capture non-linear and non-trivial relations between users and items, allowing RS to handle more complex data. Neural Collaborative

Filtering (NCF) is a CF model based on deep neural networks [He et al., 2017]. This approach combines the matrix factorization technique with neural networks in order to strengthen the performance of models based on latent factors. The architecture of NCF is shown in Figure 4.3.

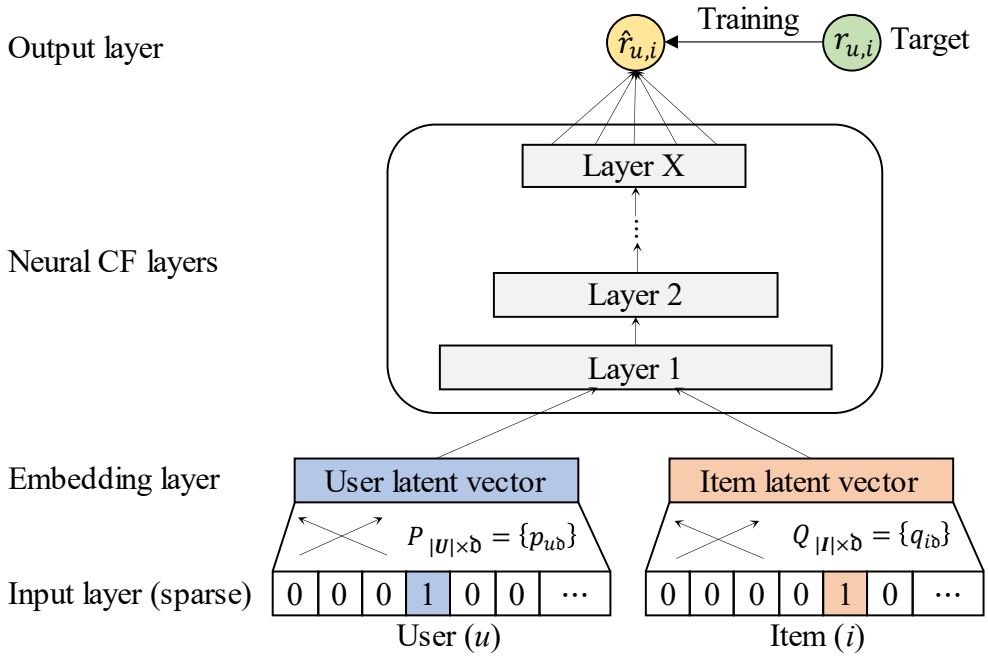


Figure 4.3: Framework of neural collaborative filtering [He et al., 2017].

The input layer consists of two sparse vectors v_u ($u \in U$) and v_i ($i \in I$) that describe user u and item i, respectively. The embedding layer is a fully connected layer that projects the sparse vectors into dense vectors (latent vectors in the context of latent factor model). The dense latent vectors are fed into a multi-layer neural network and the final output layer predicts the score $\hat{r}_{u,i}$ for target ground truth (true label). The NCF model can be formulated as equation 4.14.

$$\hat{r}_{u,i} = f(P^T v_u, Q^T v_i | P, Q, \Theta_f) \tag{4.14}$$

where $P \in \mathbb{R}^{|U| \times \vartheta}$ and $Q \in \mathbb{R}^{|I| \times \vartheta}$, denoting the latent factor matrix for users and items, respectively; and Θ_f denotes the model parameters of the interaction function f.

When handling scenarios with implicit feedback, the value of label $r_{u,i}$ is binary, 1 means item i is relevant to u, and 0 otherwise. The prediction score $\hat{r}_{u,i}$ represents how likely i is relevant to u. To endue NCF with such a probabilistic explanation, the output $\hat{r}_{u,i}$ need to be constrained in the range of [0, 1], which can be achieved by using activation function for the

output layer. [He et al., 2017] define the likelihood function as equation 4.15.

$$p(\mathscr{R}, \mathscr{R}^{-}|P, Q, \Theta_f) = \prod_{(u,i)\in\mathcal{R}} \hat{r}_{u,i} \prod_{(u,i)\in\mathcal{R}^{-}} (1 - \hat{r}_{u,i}) \tag{4.15}$$

where $\mathscr{R}$ denotes the set of observed interactions, and $\mathscr{R}^{-}$ denotes the set of negative instances, which can be all (or sampled from) unobserved interactions. Taking the negative logarithm of the likelihood, the objective function can be expressed as equation 4.16.

$$\begin{aligned} L &= -\sum_{(u,i)\in\mathscr{R}} \log \hat{r}_{u,i} - \sum_{(u,i)\in\mathscr{R}^{-}} \log(1 - \hat{r}_{u,i}) \\ &= -\sum_{(u,i)\in\mathscr{R}\cup\mathscr{R}^{-}} r_{u,i} \log \hat{r}_{u,i} + (1 - r_{u,i}) \log(1 - \hat{r}_{u,i}) \end{aligned} \tag{4.16}$$

The optimization can be done by performing Stochastic Gradient Descent (SGD) and the recommendation with implicit feedback can be addressed as a binary classification problem by employing this probabilistic treatment.

4.2.2 *Content based recommender systems*

Unlike pure CF RS only utilizes user rating matrix (either directly or via model), CB RS recommends items to a user by considering the content of the items that the user interested in the past. Various candidate items are compared with items previously rated (or used) by the user, the best-matching item(s) will be recommended [Adomavicius et al., 2005], and the content mainly provides information support in this process. The content can be different types, unstructured (e.g., textual synopsis of item) or semi-structured (a set of keywords or features of item), depending on the source of the data. To put it simply, CB RS makes recommendations by comparing the representations of item content to the representations of user interests [Melville et al., 2010].

Generally speaking, a fully functional CB RS usually contains three important phases: *i)* pre-processing; *ii)* construction of user model; *iii)* recommendation. First, having a set of data related to items, regardless of the type (structured or unstructured), CB RS analyzes the data and constructs a feature representation (generally in the form of feature vector) for each item. This phase is often called pre-processing which aims to extract relevant information to describe items. Then, the second phase builds model for the target user, which reveals user's relative interests to each of the features of the items. Finally, the third phase compares the user model with the features corresponding to the items in order to identify the items that are most relevant to the user. This phase filters items to exclude those whose features (contents) do not match to the user model.

4.2.2.1 *Pre-processing*

Pre-processing is a necessary process to extract relevant information when the input data of CB RS is raw (mainly unprocessed text). It represents the contents of items (e.g., document, Web page, and product description) in an appropriate form for the subsequent processing. The contents of items are analyzed using feature extraction techniques and preserved in a space which makes them possible to construct models. In other words, an item can be represented by a vector whose each dimension corresponds to an attribute or feature and whose each value reflects the importance (i.e., weight) of this attribute or feature for this item.

For example, in the field of document recommendation, each document is represented by a vector in a $\mathfrak{d}$-dimensional space, and the $\mathfrak{d}$ is the size of a vocabulary which is made up of the most distinguishable terms in the field and extracted from the complete collection of documents. These terms can be extracted using classic Natural Language Processing (NLP) technique, Bag-of-Words (BoW). Each dimension of the vector associated with a document represents a term from this vocabulary and the value of a dimension represents the degree of importance of the corresponding term in the description of the document content based on the usage statistics of the term in the collection.

Formally, $D = \{d_1, d_2, \cdots, d_{|D|}\}$ denotes a set of documents and $O = \{o_1, o_2, \cdots, o_{|O|}\}$ denotes a set of extracted terms. Each document $d_i \in D$ is represented by a vector with dimension $|O|$, i.e., $d_i = w_{i,1}, w_{i,2}, \cdots, w_{i,|O|}$, where $w_{i,j} (i \in \{1, 2, \cdots |D|\}, j \in \{1, 2, \cdots |O|\})$ is the weight of the term o_j in the document d_i.

The most widely used method for measuring the term weights of a document is Term Frequency-Inverse Document Frequency (TF-IDF) [Salton, 1989]. The hypothesis of TF-IDF is that the terms appeared frequently in a document (high TF) and did not appear too many times in the entire corpus are probably more relevant than others to characterize this document. The weight of a term o_j for the document d_i can be calculated by equation 4.17.

$$\begin{aligned} (a) w_{o_j,d_i} &= TF_{o_j,d_i} \times IDF_{o_j,d_i} \\ (b) TF_{o_j,d_i} &= \frac{f_{o_j}(d_i)}{f_{max}(d_i)} \\ (c) IDF_{o_j} &= \log \frac{|D|}{n_{o_j}} \end{aligned} \tag{4.17}$$

where $f_{o_j}(d_i)$ denotes the number of times the term o_j appears in document d_i, $f_{max}(d_i)$ represents the frequency of the most frequent term in d_i, and n_{o_j} denotes the number of documents in which the term o_j appears.

4.2.2.2 *Construction of user model*

After obtaining the feature vectors for each item via the first pre-processing phase, the user model can be constructed based on the given rating matrix (user ratings) and the feature vectors of the items. The construction of user model is closely related to a classification or regression in the field of machine learning. Indeed, when we treat the ratings as discrete values (like or dislike), the objective corresponds to a classification, whereas if we consider the ratings as continuous values, the objective resembles a regression.

In all cases, the set of items which the user has interacted constitutes a training process, whose ratings are the labels of the instances provided by the target user. Unlike CF, the labels of items provided by other users are excluded because each user is considered independently. In addition, we also have a set of unrated items whose labels are unknown. The training process is to construct a model for each user, and for each unrated item, the model predicts the user's interests to this item. The result presents as a score in the case of regression or the class in the case of classification.

Many classification and regression models can be used to build user model, such as K-Nearest Neighbors (KNN), decision tree, Support Vector Machine (SVM), Bayesian, and linear regression. The training process is to determine the parameters of the user model before prediction, and after that, recommendations can be made. We introduce two widely used models, KNN and linear regression.

K-nearest neighbor

KNN is an intuitive classification model. First, it defines neighbors by using a function that measures the similarity between two items. As items are represented by feature vectors, classical similarity metrics that measure the distance between two vectors are generally used, such as cosine similarity and Euclidean distance.

This similarity function is used to make predictions about items for which the active user's tastes are unknown. Formally, I_u denotes the set of items rated by the target user u and I'_u denotes the set of items not rated by u. For each item $i \in I'_u$, the set of K items (from I_u) closest to item i can be identified using the chosen similarity function. The average value of the ratings of the neighbors of i given by u is used in the KNN approach, as a prediction of $r_{u,i}$. Subsequently, the items from I'_u with the best (predicted) scores are those recommended to user u. In the case where we seek to classify item i, i.e., the ratings are discretized (ratings are not very relevant, relevant, or very relevant), we can calculate the frequency of each class among the neighbors of i and the most frequent class will correspond to the predicted label.

Linear regression

The main idea of linear regression is based on the hypothesis that there are linear relations between the ratings and the features of the items, and the ratings are considered as linear combinations of these features. In this context, the objective is to identify the weight of each feature, and use them to predict the unknown ratings by linear combinations. The training process narrows the gap between the values predicted by linear combinations and the observed rating values.

In linear regression model, the set of items rated by user u, I_u, are described as a matrix M_j, where $M_j \in |I_u| \times j$, and j denotes the number of item features. Let $\vec{r}$ denotes the column vector of dimension $|I_u|$ containing the ratings of the rated items by u. Let $\vec{w}$ be a row vector of dimension j, representing the coefficients of the features related to the rating values. Linear regression can be calculated as equation 4.18.

$$\vec{r} \approx M_j \vec{w}^T \tag{4.18}$$

To learn the parameters, the loss function is shown as equation 4.19.

$$L = \arg\min_{\vec{w}} \|M_j \vec{w}^T - \vec{r}\|^2 + \lambda \|\vec{w}\|^2 \tag{4.19}$$

where $\|\cdot\|^2$ represents the L_2 normalization of a vector and the regularization term λ is added to the parameter vector $\|\vec{w}\|^2$ to avoid overfitting.

4.2.2.3 *Recommendation*

The recommendation phase follows the user model construction phase and strongly depends on the trained user models. Given a new item representation, the recommendation phase predicts whether it is likely to be one of the interest for the target user, by comparing features in the item representation to those in the representation of user interests (stored in the user model) [Lops et al., 2011]. Usually, the recommendation phase implements some strategies to rank potentially interesting items based on the relevance to the user model, i.e., top-K recommendation.

Besides, user interests usually change in time, therefore, new information should be maintained and provided to the user model construction phase, in order to automatically update the user models. Further feedback is gathered on generated recommendations by letting users state their satisfaction or dissatisfaction with recommended items. After gathering that feedback, the model construction process is performed again on the new training data, and the results about interests are adapted to the user models. The iteration of the feedback-updating cycle over time allows the system to take into account the dynamic information of user interests [Tang et al., 2021].

4.2.3 *Hybrid recommender systems*

Hybrid RS emerged to compensate for the flaws of CF RS or CB RS. Before presenting the hybrid RS, we highlight the flaws of CF RS and CB RS.

CF RS has two main flaws: data sparsity and cold start problem [Balabanović et al., 1997].

Data sparsity. Reliable recommendations usually require a huge amount of ratings for CF RS, but scenarios with small data are also very common. In such scenarios, the size of samples (users and items) is large, but there are a lot of null values in the rating matrix that are either missing or never appear. In fact, sparse data does not mean that the data is useless, just that the information is incomplete and a lot of useful information can be mined through appropriate means.

Cold start problem. The problem related to new user or new item is called cold start. The RS may have difficulty recommending new items to users or generating recommendations for new users. CF RSs particularly suffer from this problem. When a new user or new item present in the domain, the RS does not have any information (previous ratings) about the new ones, thus, CF can not be performed in such context.

CB RS also has cold start problem as well as two other main flaws: content analysis limitation and overspecialization [Balabanović et al., 1997]. The CB RS can not precisely capture the interests of a user in relation to the features of items and cannot build reliable model if the user rated very few items, because insufficient data makes the process of building user model unreliable.

Content analysis limitation. CB RS requires the features associated with items. Therefore, in order to obtain sufficient features, the item content should be in the form that can be parsed automatically by computer or the features can be assigned to items manually [Adomavicius et al., 2005]. Although some information retrieval techniques work well in extracting features from text documents, some other domains have feature extraction problem [Adomavicius et al., 2005], such as multimedia with image, audio, and video. Moreover, it is not practical to assign attributes by hand due to amount of resources [Shardanand et al., 1995]. Another problem of content analysis limitation is that if two different items are represented by the same set of features, they are indistinguishable. For example, texts are usually represented by their most important keywords, CB RS cannot distinguish the quality of two papers shared the same keywords.

Overspecialization. CB RS may fall into a situation where it only recommends items that are highly relevant to the user's profile, and the user only receives the similar items in limited range. For example, a person with no feature about computer science would never receive a recommendation for even the booming ChatGPT. This problem, which has also been studied in other domains, is often addressed by introducing some randomness (e.g.,

genetic algorithms) [Sheth et al., 1993]. And in some certain cases, items should not be recommended if they are too similar to something the user has already used (e.g., a different news describing the same event). In summary, the diversity should be considered, user should be presented with a range of alternatives, instead of a set of homogeneous items.

To overcome the main flaws of the above two approaches and take advantage of their strengths, hybrid RS has emerged in the literature. Different hybrid RSs have been proposed with the objective of combining multiple recommendation approaches [Aggarwal et al., 2016a]. Some widely used hybrid approaches can be classified into four types: separate recommenders, adding content features to CF, adding collaborative features to CB, and unified recommendation model.

Separate recommenders. This approach implements CB RS and CF RS separately, then combines their predictions. In such context, we can combine the predictions obtained from two RSs (CF and CB) into one final recommendation using linear combination or voting.

Adding content features to CF. This approach adds some content features in CF RS. It builds hybrid RS based on conventional CF algorithm but also maintains the user profile for each user, the profiles are usually used to calculate the similarities between users. This approach can overcome the data sparsity problem of conventional CF RS.

Adding collaborative features to CB. This approach adds some collaborative features in CB RS. It builds hybrid RS based on conventional CB algorithm but also uses some dimension reduction techniques on the set of user feature matrix obtained from user profiles.

Unified recommendation model. This approach constructs unified model that incorporates both content and collaborative features. [Popescul et al., 2013] propose an unified probabilistic method for combining CB and CF approaches, which is based on latent semantic analysis [Hofmann, 1999]. Representatively, [Ansari et al., 2000] compress user profile and item content in a single statistical model that estimates unknown ratings $r_{u,i}$ between them.

Hybrid RS can also be augmented by knowledge-based techniques in order to improve recommendation accuracy and to address some of the limitations (e.g., cold start problem) of traditional RSs [Burke, 2000]. In summary, a hybrid RS theoretically performs better than a single RS, since it considers more information, combines the advantages of the individuals, and overcomes their flaws. Moreover, several papers compare the performance of the hybrid RSs with pure CF or CB RSs, and demonstrate that hybrid approaches can provide more accurate recommendations than single approaches [Balabanović et al., 1997] [Pazzani, 1999] [Melville et al., 2002].

4.3 EVALUATION OF RECOMMENDER SYSTEMS

The application scenarios of RSs can be mainly categorized into two groups: rating prediction and top-K recommendation [Herlocker et al., 2004] [Cremonesi et al., 2010] [Steck, 2013] [Gunawardana et al., 2012]. In this section, we present evaluation metrics used in the above two groups of application scenarios.

In the context of evaluation based on rating prediction, it considers that the accuracy of RSs is compared based on their ability to correctly predict users' rating values. As ratings are generally represented by numerical values, regression models are usually considered. $r_{u,i}$ is a real rating of user u for item i; $\hat{r}_{u,i}$ is a predicted rating of user u for item i; S_{test} is subset of the ratings used to evaluate the RS.

Root Mean Squared Error (RMSE) (equation 4.20) is frequently used to evaluate the overall prediction error, small RMSE values indicate better prediction performance, and it is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in S_{test}} (r_{u,i} - \hat{r}_{u,i})^2}{|S_{test}|}} \tag{4.20}$$

RMSE tends to disproportionately penalize large errors due to the squared term in the sum [Aggarwal et al., 2016b]. Another metric Mean Absolute Error (MAE) (equation 4.21) does not suffer from this defect, it measures the average magnitude of differences between prediction and real ratings [Willmott et al., 2005], and it is defined as follows:

$$MAE = \frac{\sum_{(u,i)\in S_{test}} |r_{u,i} - \hat{r}_{u,i}|}{|S_{test}|} \tag{4.21}$$

Generally, RMSE is more sensitive with errors in predictions compare to MAE and therefore the RS will be considerably penalized by a few poorly predicted ratings, but it does not accurately reflect the average error that may lead to confusing results [Willmott et al., 2005].

In the context of evaluation based on top-K recommendation, it considers the items presented in lists and usually return a recommendation list with a K length. $S_{test}(u)$ denotes the set of ground truth items with u in the test subset, where the relevant items for a user can be the items used (or rated) by the user (implicit feedback) or items whose rating values exceed a threshold (explicit feedback). $L_{rec}(u)$ denotes the item list recommended to u, and $|L_{rec}(u)| = K$.

Precision (equation 4.22) and Recall (equation 4.23) are widely used in top-K recommendation scenarios for performance evaluation [Liu et al., 2016] [Zhou et al., 2018] [Bobadilla et al., 2020]. Precision counts the proportion of items relevant to user u among the recommendation list and

Recall focuses on the percentage of relevant items among the ground truth items.

$$\text{Precision} = \frac{|S_{test}(u) \cap L_{rec}(u)|}{K} \tag{4.22}$$

$$\text{Recall} = \frac{|S_{test}(u) \cap L_{rec}(u)|}{|S_{test}(u)|} \tag{4.23}$$

F1 (equation 4.24) takes the harmonic mean of Precision and Recall, it is defined as follows:

$$F1 = \frac{2}{\text{Precision}^{-1} + \text{Recall}^{-1}} \tag{4.24}$$

The above three evaluation metrics do not take into account the order of the items in the list. But RS should put the relevant items in the head of the recommendation list. Thus, the order of the list should be considered. Mean Average Precision (MAP) (equation 4.25) and Normalized Discounted Cumulative Gain (NDCG) (equation 4.26) give recommendation lists in which the most relevant items are presented at the top of the list. MAP represents the average value of all Average Precision (AP) values over all users.

$$MAP = \frac{1}{|S_{test}|} \sum_{u=1}^{|S_{test}|} \frac{1}{|S_{test}(u)|} \sum_{k=1}^{K} \text{Precision} \tag{4.25}$$

NDCG counts the distance between the top-K recommendation list and ground truth list. $rel_{u,i}$ denotes the relevance of item i for user u; $rel_{u,i}$ equals to the rating value $r_{u,i}$ in the context of explicit feedback and $rel_{u,i} = 0$ (there is intersection between u and i) or $rel_{u,i}$ (there is not intersection between u and i) in the context of implicit feedback; $index_i$ denotes the position of item i in the recommendation list; IDCG denotes the ideal recommendation list, i.e., all the items are ordered in the list according to the ground truth list.

$$NDCG(L_{rec}(u)) = \frac{1}{IDCG(L_{rec}(u))} \sum_{i=1}^{K} \frac{2^{rel_{u,i}} - 1}{\log_2(index_i + 1)} \tag{4.26}$$

Hit Rate (HR) (equation 4.27) is also widely used in top-K recommendation scenarios, but $K = 1$, i.e., it considers a single target item for the user u. The hit will be confirmed only if the target item is present in the recommendation list. The value of $\mathbb{I}(\cdot)$ is 1 when $\cdot \geqslant 0$, and 0 otherwise.

$$HR = \frac{\sum_{u \in S_{test}} \mathbb{I}(|S_{test}(u) \cap L_{rec}(u)|)}{|S_{test}|} \tag{4.27}$$

4.4 KNOWLEDGE GRAPH IN RECOMMENDER SYSTEM

Although numerous efforts have been made toward more personalized recommendations, RSs still suffer from several challenges, such as data sparsity and cold-start problem [Guo et al., 2020]. In recent years, building RSs with KG as side information has garnered enough attention. With the help of KG, the latent connections between users and items can be obtained [Wang et al., 2018a], such an approach can not only alleviate the above mentioned challenges to get more accurate recommendations, but also provide explanations for the recommended items [Guo et al., 2020].

Embedding based method is the representative direction of the usage of KG in RSs, which leverages facts of KG to enrich the representations of users and items. It contains two modules: *i)* graph embedding module learns the representations of entities and relations in the KG; *ii)* recommendation module estimates users' preferences for items with learned features. The challenges of this method are how to obtain the entity embedding with proper KGE algorithm and how to integrate the learned entity embedding in the recommendation module.

Based on the structure of the combination of these two modules, embedding based method can be subdivided into three categories: two-stage learning, joint learning, and multi-task learning [Guo et al., 2020].

Two-stage learning

Two-stage learning trains graph embedding module and recommendation module respectively. First, embeddings of entities and relations are learned with KGE algorithms. Then, the related embeddings are fed into the recommendation module along with other user features and item features to make predictions. For example, [Wang et al., 2018b] propose Deep Knowledge-aware Network (DKN) for news recommendation. DKN extracts entities from news titles and maps them into KG to mine the knowledge-level relations between news. It models the news p_j by combining the textual embedding of sentences learned with Convolutional Neural Network (CNN) [Kim, 2014] and the knowledge-level embedding of entities in news content via TransD [Ji et al., 2015] to get the final news representation v_{p_j}. In order to capture user's dynamic interests in news, the representation v_{u_i} of user u_i is learned by aggregating the embedding of historical clicked news $\{p_1, p_2, \cdots, p_n\}$ with an attention mechanism, shown as equation 4.28.

$$v_{u_i} = \sum_{k=1}^{n} s_{p_k,p_j} v_{p_k} \tag{4.28}$$

where s_{p_k,p_j} measures the similarity between the candidate news p_j and the clicked news p_k. Then, user's preference for candidate news p_j can be calculated via $\hat{r}_{(u_i,p_j)} = \text{MLP}(v_{u_i}, v_{p_j})$.

Two-stage learning is easy to implement because the structure embeddings of KG are joined as extra features in the subsequent recommendation module. Moreover, there is no need to update the embeddings frequently once they are learned since the KG is usually stable. However, the structure embeddings of KG learned by KGE algorithms are more apposite for internal graph applications (e.g., KG completion) because embedding module and recommendation module are relatively independent [Guo et al., 2020].

Joint learning

Joint learning jointly learns the graph embedding module and the recommendation module in an end-to-end way. In this way, the recommendation module can guide the feature learning process in the graph embedding module. For example, [Zhang et al., 2016] propose Collaborative Knowledge base Embedding (CKE). It unifies various types of side information in a CF based framework, including attribute feature, textual feature, and visual feature of items. The attribute feature is encoded with by TransR [Lin et al., 2015] to learn structure embedding of KG; the textual feature and the visual feature are extracted with autoencoder. The objective function of these three feature learning modules are added with the recommendation module to learn parameters jointly, shown as equation 4.29.

$$\mathcal{L} = \mathcal{L}_{\text{Rec}} + \lambda_1 \mathcal{L}_{\text{attribute}} + \lambda_2 \mathcal{L}_{\text{text}} + \lambda_3 \mathcal{L}_{\text{visual}} + \lambda_4 \mathcal{L}_{\text{regu}} \tag{4.29}$$

where $\mathcal{L}_{\text{Rec}}$, $\mathcal{L}_{\text{attribute}}$, $\mathcal{L}_{\text{text}}$, $\mathcal{L}_{\text{visual}}$, and $\mathcal{L}_{\text{regu}}$ are the objective function of the recommendation module, the attribute feature learning module, textual feature learning module, visual feature learning module, and the regularization term, respectively. The final representation of item v_{p_j} is obtained by aggregating the item feature from each part. After obtaining the latent vector v_{u_i} of the user u_i, the preference score between user u_i and item p_j is estimated via the inner product $\hat{r}_{u_i,p_j} = v_{u_i}^{\top} v_{p_j}$.

The joint learning can be trained end-to-end, and it uses KG structure to regularize the RS, the experiments show that incorporating structural knowledge can improve the performance of recommendation [Guo et al., 2020]. However, the combination of different objective functions needs to be fine-tuned, which increases the computational complexity.

Multi-task learning

Multi-task learning trains the RS with the guidance of the KG-related task. The motivation is that the structure of the items in the user-item interactions is similar to the structure of users and items in the KG. Therefore, the transferring of features of items and users from two sources (interactions and KG) can help improve the performance of RS. [Wang et al.,

2019a] propose multi-task feature learning for knowledge graph enhanced recommendation, which consists of a recommendation module and a KGE module. Instead of feeding graph structure embeddings into the recommendation module, the two modules are independent and connected with a cross&compress component to share knowledge with each other. The KGE module is trained to estimate feature representation of the tail given the head entity and the relation in a triplet $\langle h, r, t \rangle$, and the recommendation module is trained to make predictions by integrating prominent features. [Cao et al., 2019] propose to learn the tasks of both recommendation and KG completion at the same time. Since user preferences can be reflected by relations among items and some facts are missing in the KG, the feature representations learned from recommendation task can be transferred to the KG completion task to improve the KG; on the other hand, better feature representations of users and items can be obtained form the improved KG and used in the recommendation task.

Multi-task learning can prevent RS from overfitting and improve the generalization ability of the model. However, similar to the joint learning method, it requires efforts to integrate different tasks under one recommendation framework [Guo et al., 2020].

4.5 RECOMMENDER SYSTEM IN E-LEARNING

E-learning offers spatially independent education platforms with a large number of high-quality and convenient online resources that have the potential to improve educational outcomes for users around the world. However, the explosive growth of Web information makes it difficult for users to make efficient choices. RS has become an effective solution to extract and organize the learning content to achieve the established learning objectives of users, especially for non-experts [Shi et al., 2020].

The applications of RS in e-learning are extensive, the recommended content is not just limited to learning material, it also includes the recommendations regarding learning path [Shi et al., 2020], course [Zhang et al., 2023], collaborator [Liu et al., 2018], etc. Many techniques are found in the literature of this domain, in addition to the three (CF, CB, and hybrid) we have presented above, others such as knowledge based RS [Burke, 2000], context based RS [Adomavicius et al., 2010], and KG based RS [Guo et al., 2020]. Different techniques may construct different architectures of RSs, but like the standard version, a qualified e-learning RS contains three components: user interface, database server,and recommendation engine. User interface is responsible for generating requests and handling logic events; database server stores all the data of RS, including user profiles, resource contents, and activity records; recommendation engine implements the algorithm and generates recommendations for users.

[Shu et al., 2018] proposed a content-based recommendation algorithm for learning resources. This algorithm can be summed up as two processes, training process and recommendation process. In the training process, a convolutional neural network is constructed and used to obtain features from text information. In the recommendation process, user preferences combined with the obtain features are used to predict ratings.

As MOOCs occupy an increasingly important proportion in e-learning, the course recommendation of MOOCs has received considerable attention, and many scholars have contributed to this domain [Jing et al., 2017] [Zhang et al., 2019a] [Lin et al., 2021] [Hao et al., 2023]. Course recommendation has its own characteristics. First, the knowledge concepts between courses are closely related, displaying complex semantic collections, such as the prerequisites of courses. Second, users have particular learning objectives and activities performed according to these objectives, so it is necessary to take the learning objectives and future plans into consideration.

[Chen et al., 2021] proposed a learning path recommendation framework for MOOC platforms based on KG. Based on the current course situation of MOOC platforms, the authors propose a new automated construction method for course KGs. A course KG is constructed by annotating the pre-knowledge of each course and calculating the similarity between courses, and it is displayed using the Neo4j graph database platform. After completion of the course KG, it is used to study learning path recommendation algorithms, including rule-based and machine learning based algorithms, and to perform a comparative analysis using the higher education formation program of a university.

There are three main steps for a course RS function appropriately: data collection, feature extraction, and recommendation [Liu et al., 2022]. The data collection acquires as much information as possible that the RS needs, including user related information and course related information. The feature extraction obtains user features (explicit features form user profile and implicit features from user activity) and course features (explicit features form course content) from the data. In this step, data mining technology and embedding technology is mainly used, may accompanied by machine learning or deep learning. The third step is recommendation, which mainly consists of two stages: the recall stage and the ranking stage. The recall stage uses several algorithms and models to recall courses that may be interested to users from a large amount of alternatives. The ranking stage estimates user preferences from the set of items of the recall layer and make recommendations. The overall framework of course recommendation is shown in Figure 4.4.

In addition, some strategies and algorithms (complementary strategy) can be added to update the recommendation list before returning it to the

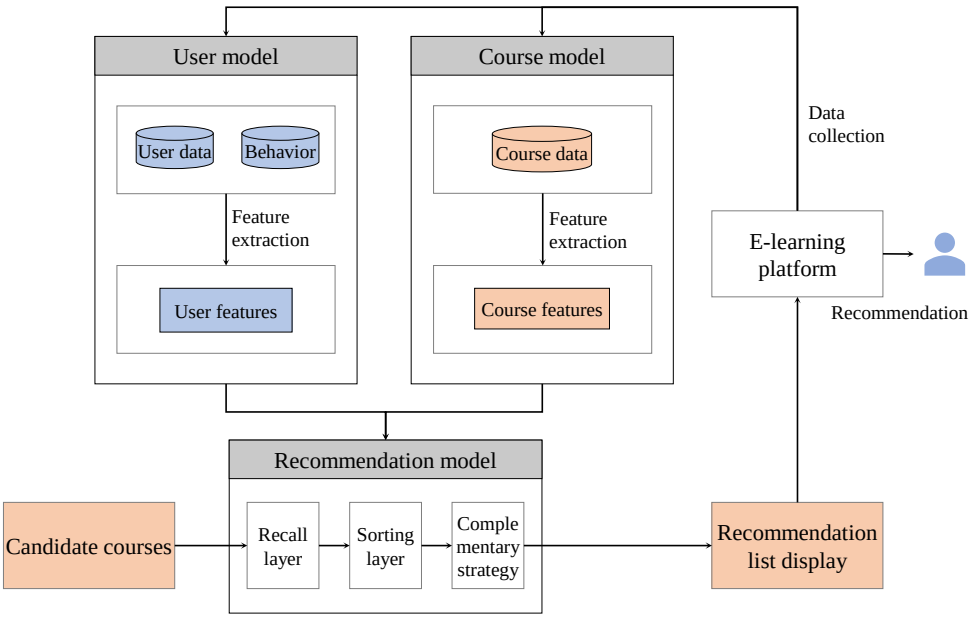


Figure 4.4: Overall framework of course recommendation [Liu et al., 2022].

user, such as context-aware and knowledge based supports, taking into account other conditions [Liu et al., 2022].

4.6 CHAPTER SUMMARY

In this chapter, we presented the technologies related to RS. We first introduced the information used in RSs. Then, we presented three categories of RSs and highlighted their corresponding flaws. Next, we listed several evaluation metrics used for measuring the effectiveness of RSs and different recommendation scenarios may require different metrics. Later, we introduced the applications of KG in RSs. Finally, we briefly introduced the applications of RS in e-learning, the course recommendation of MOOCs platforms was illustrated as a representative case.

As an important tool to help users filter information, RS has been applied to many fields, and the e-learning field is no exception. However, most of the conventional RSs ignore a piece of crucial information, the latent connections between the entities in the specific domain, which can be a part of the basis for the RSs to make judgments. To capture the latent connections, we plan to integrate the technologies introduced above to form a recommendation framework. The framework uses KG to characterize the content of e-learning platform and model the information to obtain more valuable features to improve the performance of the downstream RS. The specific details of the framework will be introduced in the next chapter.

Part III

CONTRIBUTIONS

Chapter 5

Recommendation framework based on knowledge graph

In this chapter, we introduce our contribution related to the design of a Recommendation Framework based on Knowledge Graph (RFKG) in e-learning platforms. Before we detail the methodology of RFKG, we highlight the research context and questions.

> **Context.** RSs are applied in e-learning platforms to recommend pedagogical resources to users. But conventional recommendation approaches cannot fully extract information from platforms, where the missing information can provide crucial support to RSs. KG can characterize the content of e-learning platform in the form of machine readable graph and comprehensively capture valuable information, both explicit and implicit, to support the downstream RS.
>
> **Questions.** How to characterize the content of e-learning platform by the means of KG? Can the KG be generic? How to obtain features of users and resources from the constructed KG? How to learn these features with algorithm for recommendation?

The organization of this chapter is as follows. Section 5.1 presents the framework overview, including its architecture and brief introduction. Section 5.2, section 5.3, and section 5.4 detail the three main modules of the framework. Section 5.5 is a summary of this chapter.

5.1 Overview

A recommendation framework is a set of layered functions that are available to software developers to be used as tools for the development of RS [Sielis et al., 2015], and we call these layered functions modules. To construct a recommendation framework for e-learning platforms, the basic components and the challenges we need to address are shown in Figure 5.1.

First, in e-learning platforms, the data that can be collected and used is usually categorized into explicit data and implicit data. Among them, the

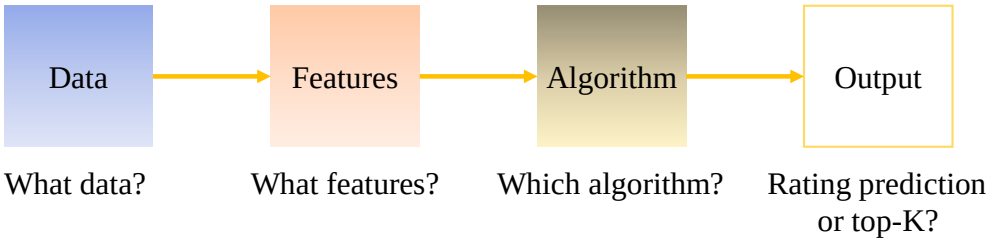


Figure 5.1: Basic components and challenges of recommendation framework.

explicit data includes user ratings, user profile, resource content, etc. Most of the explicit data are text-based. While the implicit data includes user browsing records, user interaction records with resources (completed or dropout), semantic information, etc. Second, in any recommendation scenario, the entity classes primarily considered by RS are user and resource. Obtaining features of these two classes of entities from their instances is a prerequisite for the RS to function properly and these features should be able to capture the commonalities and differences between users and resources. Third, the core engine of a RS necessarily contains one or more algorithms, the algorithms make up the brain of RS, which learns the features of users and resources and makes assertions about unknown users and resources based on the learned experience. Finally, in the e-learning recommendation scenario, the RS usually does not predict the user ratings of resources (as Netflix does), but selects the resources that are most likely to be of interest to the user from a pool of alternatives, i.e., top-K recommendations.

Based on the analysis above, we propose RFKG, a recommendation framework that assists users in selecting appropriate pedagogical resources in e-learning platforms. It contains three modules:

i) **Graph construction** characterizes the content of e-learning platform via KG;

ii) **Feature extraction** acquires the features of users and resources from the constructed KG;

iii) **Recommendation** learns the acquired features with MLP for resource recommendation.

The architecture of RFKG is shown in Figure 5.2. In the graph construction module, we define entity classes and their relation classes based on the characteristics of e-learning platform, extract instances of these entity

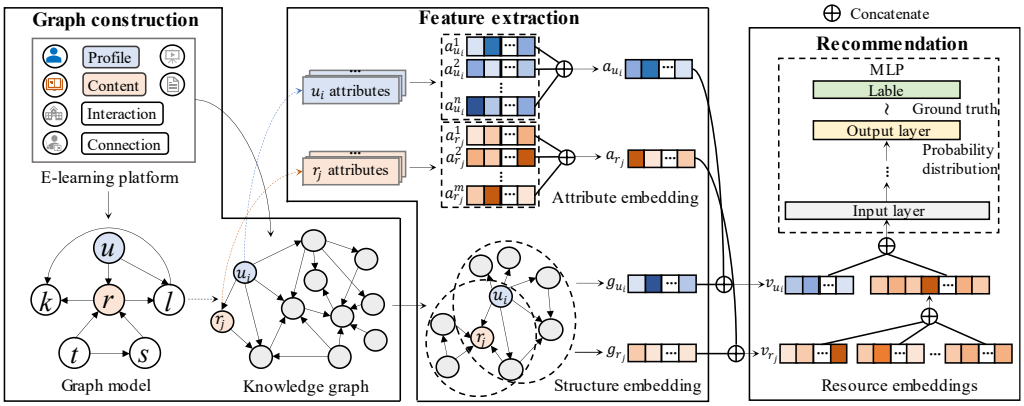


Figure 5.2: Architecture of RFKG.

classes along with their attributes from the raw platform data, and compressed them into a KG in accordance with the defined relation classes. In the feature extraction module, we choose two embedding methods to learn the two types of information (textual attributes and graphic structure) contained in the constructed KG to obtain textual and structural features of users and resources. In the recommendation module, we concatenate the user and resource feature vectors learned through feature extraction to form the final user and resource feature vectors, respectively. These feature vectors and user-resource interaction records are then fed into the MLP network for model fitting, score prediction, and resource recommendation.

5.2 GRAPH CONSTRUCTION

In this section, we construct a lightweight ontology represented as a KG following 4 core steps [Noy et al., 2001]: *step 1.* Determine the domain and scope of ontology; *step 2.* Define the entity classes and their relation classes; *step 3.* Define the classes of attribute associated with entity classes; *step 4.* Extract the instances of the defined classes and compressed them into a knowledge base represented as a KG.

The domain of KG is e-learning platform and the scope is resource recommendations to users. To simplify the construction process, we merge *step 2.* and *step 3.* into one step, **terminology**, which defines a conceptual graph model made of multiple classes of entity, relation, and attribute. The conceptual graph model is a theoretical mold used as the rules of importing the instances of these classes into KG. Then, we name the *step 4.* **assertion**, which extracts and compresses the instances of the defined classes into KG. Thus, the process of the first module (graph construction) can be summarized into the following two components:

1) **Terminology** defines a conceptual graph model, involving the classes of entity, relation, and attribute, as a domain vocabulary;

2) **Assertion** extracts the instances of the defined classes and imports these facts into KG according to the domain vocabulary.

5.2.1 *Terminology*

The input to the terminology component is raw data from the application domain (e-learning platform) and the output is a conceptual graph model. A graph model can be described as: $M = \{\mathbf{E}, \mathbf{R}, \mathbf{A}\}$, where **E**, **R**, and **A** are sets of entity classes, relation classes, and attribute classes, respectively, and they are defined according to the characteristics of e-learning platform. To achieve that, we need to analyze the mechanisms of typical e-learning platforms and figure out what kind of raw data we can obtain from the platforms as input to this component.

As we presented in chapter 2, most of the e-learning platforms are Web-based and user-oriented, such as Blackboard, Desire2Learn, ILIAS, Moodle, Sakai, and the platforms of MOOCs. In these application scenarios, platforms are service providers and users are service consumers. We can identify that there are two different sources of data in a typical Web-based and user-oriented e-learning platform, the data not related to any user and the data generated by users. We name the two sources of data **knowledge base** and **user-related data**, respectively.

- **Knowledge base.** The knowledge base of e-learning platform is not related to any user, it refers to any kind of information support and its participants. For example, in Coursera, the knowledge base includes courses, course synopses, teachers, teacher profiles and avatars, institutions, institution synopses and icons, and some other supplementary information. In other words, knowledge base refers to the data that exists before any user registers the platform;

- **User-related data.** User-related data is generated by users themselves, and it mainly contains two types of data: user profile and user activity. As presented in section 2.3, user profile is a collection of personal information, which is stored without adding further description or interpretation, such as background (e.g., name, gender, degree, profession, age, and spoken language), preference (e.g., language preference and resource provider preference), and competency (e.g., level of English speaking and proficiency of python). On the other hand, users perform various user-centered learning activities with specific learning purposes in the e-learning platforms, the records of these activities can be stored in the form of Web logs and these logs can be collected by service providers (i.e., platforms) to

improve the experience of service consumers (i.e., users) by incorporating RSs.

After clarifying the two sources of data, we need to extract the terminologies from them, that is, the elements (classes of entity, relation, and attribute) of the conceptual graph model.

5.2.1.1 *Entity classes and attribute classes*

An e-learning platform contains numerous entities, intricate relations between them, and massive amounts of additional information. But no matter what platform, user and resource are two essential entity classes. We select the entity classes directly related to them from other possible entity classes. Then, among the attribute classes that these selected entity classes may contain, we select only the relevant ones that may affect the user's choice as attribute classes in the conceptual graph model.

User

Users refer to millions of people around the world who use e-learning platforms to learn for a variety of reasons, including career development, changing careers, college preparations, supplemental learning, lifelong learning, corporate e-learning & training, etc. Users have to register in order to get access to e-learning platforms. During registration, users will have to fill in some of the necessary information, which will be saved in the user profiles. Figure 5.3 is an example of user profile in edX. The information may include username, e-mail address, degree, gender, major, spoken language, preference, experience, skill, location, etc.

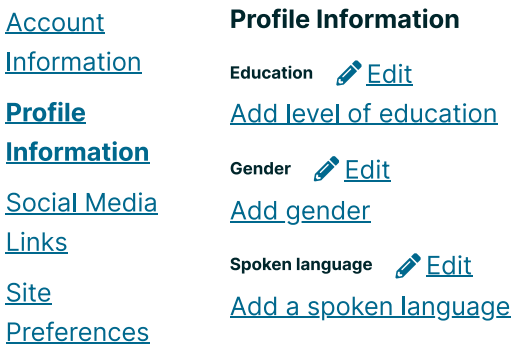


Figure 5.3: Example of user profile in edX.

We create a user model to store the information of user profile [Tang et al., 2022]. The information composition of the created user model is shown in Table 5.1. Note that users may have more information than what is shown in the table, and the model allows to add or delete elements according to specific platforms. The information of user model contains a large number of valuable features, which is exactly what the RS needs.

Table 5.1: User model [Tang et al., 2022].

Information	Component	Value
Identity	Username	#string
	Age	$\mathbb{N}^+$
	Gender	male, female
	Degree	engineer, master, doctor, etc.
	Major	mechanics, biology, chemistry, etc.
	...	...
Preference	Format	document, audio, video, etc.
	Language	English, French, Spanish, etc.
	...	...
Competency	English	A1, A2, B1, B2, C1, etc.
	Programming	beginner, advanced, expert
	Machine learning	beginner, advanced, expert
	...	...

So, the user model can be attached as an attribute class to user in the conceptual graph model.

Resource

Similar to online shopping, resources are products to e-learning platforms. Generally, in a user-oriented learning platform, there are a large number of resources, organized by subject or field, and users can navigate or search for resources, select resources to learn according to their needs. Resource is usually provided with a brief synopsis or description in the platform to give users a general understanding of the resource before they select it. For example, in MOOCs, courses serve as resources, and the course sponsor usually attaches a short introductory paragraph to the course, in the form of a synopsis. Figure 5.4 is an example of course synopsis in edX.

The textual synopsis contains a large amount of information about the resource, and features about the resource can be extracted from this information which can be used in recommendation tasks. Thus, the synopsis can be defined as an attribute class of resource in the conceptual graph model.

School

As mentioned in section 2.3, the development of e-learning has been heading forward to the direction of standardization. In user-oriented e-learning platforms, resources are usually provided by institutions, such as universities, companies, and specialized training institutions. These institutions function similarly to schools, and usually partner with the platforms. Generally speaking, the better the reputation of a school, the

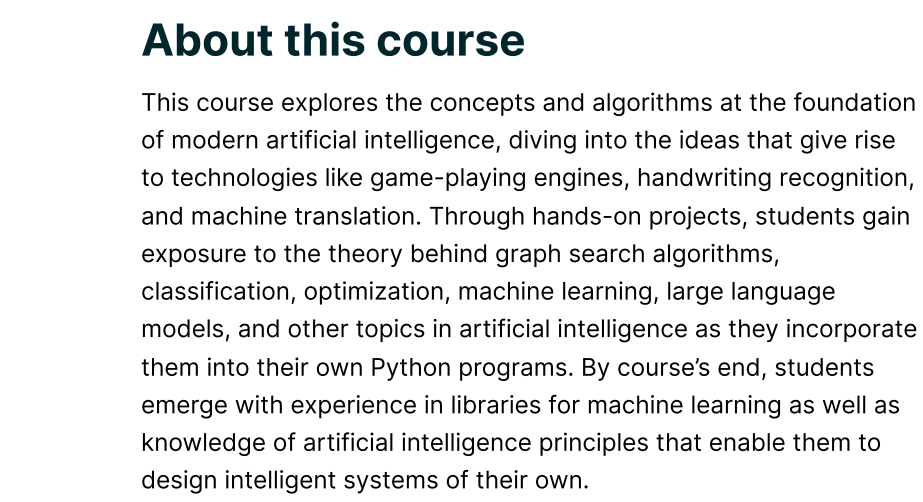


Figure 5.4: Example of course synopsis in edX.

more credible the resources launched by that school will be. Similarly, school is usually accompanied by a short synopsis, as shown in Figure 5.5.

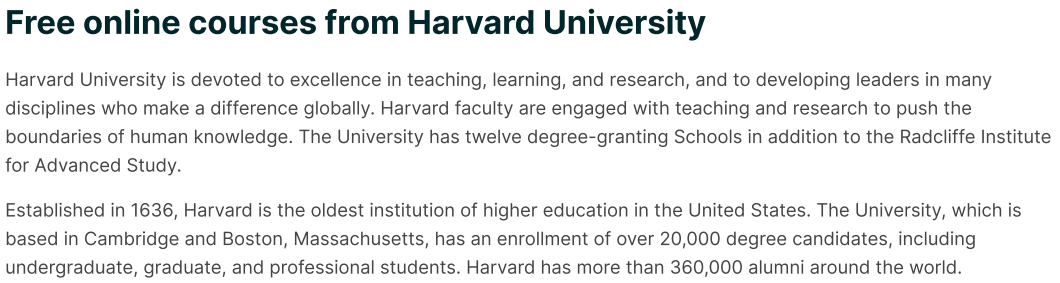


Figure 5.5: Example of school synopsis in edX.

Teacher

Teacher refers to the instructor of resource. Teachers usually have partnerships with platforms or schools, and they teach their resources through the platforms. Over time, different users may have their own preferred teachers, and the resources directed by these teachers may become their priorities. Similarly, for teachers who are not well known, attractive synopsis is an effective channel for users to get to know them. Thus, teacher also has a synopsis as attribute, which mainly includes work unit, education background, subject, and teaching experience.

Learning material

Learning material refers to all learning-related content contained in resource, such as video, practice exercises, reading, quiz, exam, etc. In user-oriented e-learning platforms, resource sometimes is not just single individual. For example, in Coursera, the resource is course, which often contains many sub-contents, such as video lectures, readings, and quizzes.

Formally, learning material is the smallest individual of the sub-content in resource.

Knowledge concept

Knowledge concept refers to the basic unit that conveys pedagogical information in the process of learning activities, including theory, principle, definition, algorithm, example, conclusion, etc. User usually aims to acquire or improve the ability of certain knowledge concepts. In other words, knowledge concept is the skills or competencies related to certain knowledge domains. An example of knowledge concepts is shown in Figure 5.6.

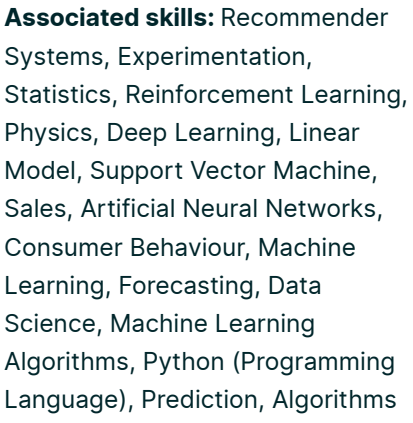


Figure 5.6: Example of knowledge concept in edX.

5.2.1.2 *Relation classes*

After the entity classes and attribute classes are extracted, the relation classes between these entity classes can be defined. They are summarized as:

$\mathit{User} \xrightarrow{\mathrm{select}} \mathit{Resource}$: Users select resources according to their learning objectives;

$\mathit{Resource} \xrightarrow{\mathrm{contain}} \mathit{Learning\ material}$: Generally, a resource is a collection of learning materials, such as video lectures and documents;

$\mathit{User} \xrightarrow{\mathrm{use}} \mathit{Learning\ material}$: User will be granted access to use the learning materials included in the resource after selecting a resource;

$\mathit{Teacher} \xrightarrow{\mathrm{teach}} \mathit{Resource}$: Resource is taught by one or more teachers;

$\mathit{School} \xrightarrow{\mathrm{offer}} \mathit{Resource}$: In commercial e-learning platforms, resources are usually provided by schools that partner with the platforms to ensure the reliability. Of course, in community-like e-learning platforms, resources can also be provided by individuals;

Teacher $\xrightarrow{\text{in}}$ *School*: Generally, teacher is affiliated with school;

Resource $\xrightarrow{\text{refer to}}$ *Knowledge concept*: Resource refers to (i.e., involves) one or more knowledge concepts, and they are important considerations for users when selecting resources;

Learning material $\xrightarrow{\text{refer to}}$ *Knowledge concept*: Learning material refers to (i.e., involves) one or more knowledge concepts.

5.2.1.3 *Graph model*

Based on the defined classes of entity, relation, and attribute, we can design the conceptual graph model, shown in the Figure 5.7. The model is adaptive, i.e., the defined classes can be adjusted according to the specific e-learning platform.

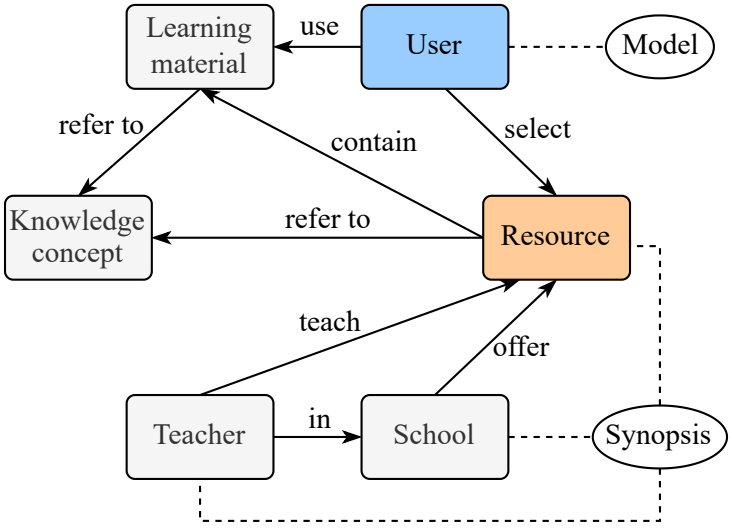


Figure 5.7: Designed conceptual graph model.

5.2.2 *Assertion*

The process from terminology component to assertion component is from conceptual graph model to formed KG. After the graph model is designed, the instances of classes (entity, relation, and attribute) can be extracted from the data of e-learning platform and imported into a KG according to the graph model. For example, we populate a course of Coursera, named *Unsupervised Learning, Recommenders, Reinforcement Learning*, into KG. The part of the formed KG is shown in Figure 5.8.

The assertion component is a batch import process, from data to graph database. To do that, manual import can be used when the amount of data

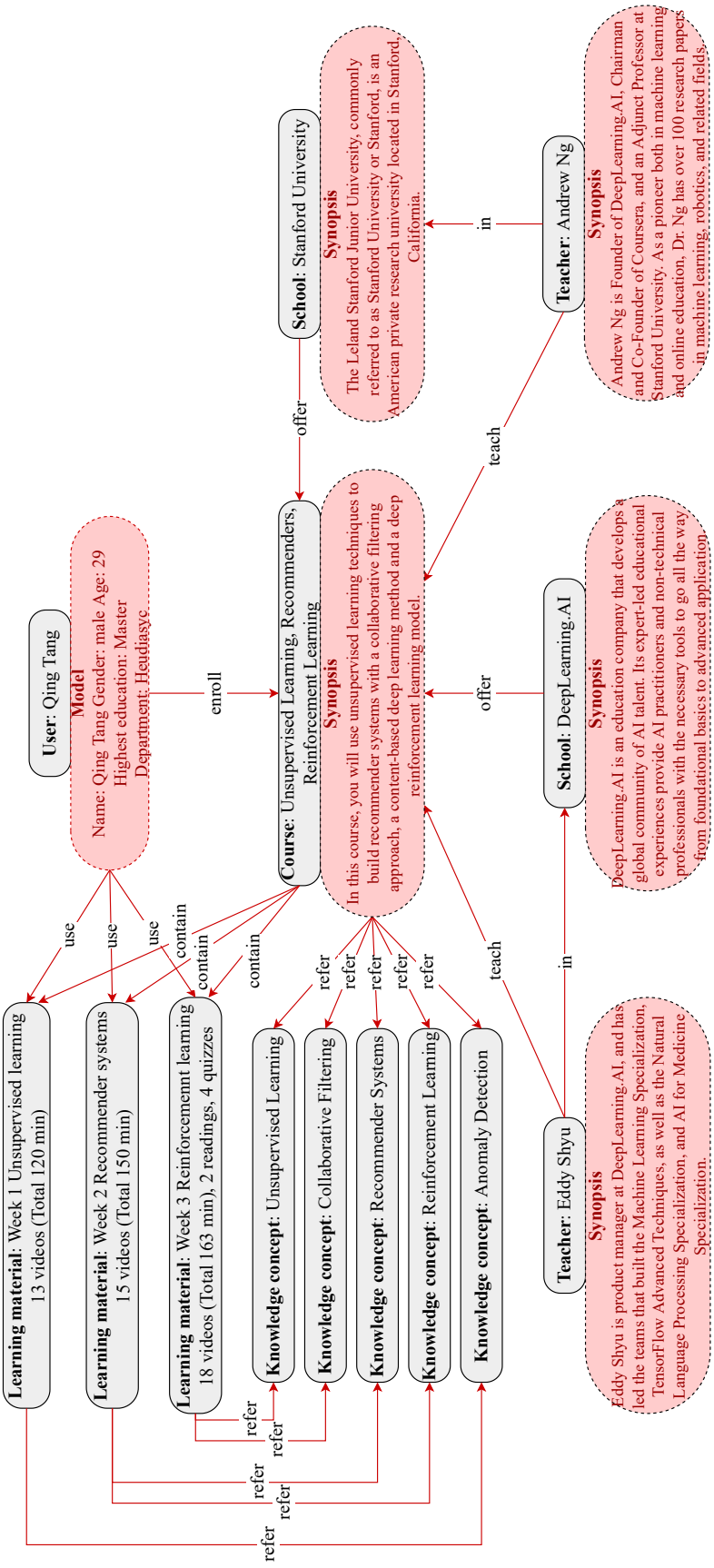


Figure 5.8: Illustration of the part of constructed KG.

is small, but bulk import with the API provided by the graph database (e.g., Neo4j) is usually the first choice when the amount of data is large.

Formally, a formed KG can be described as $G = \{(\mathbf{e}_u, \mathbf{e}_r, \mathbf{e}_t, \mathbf{e}_s, \mathbf{e}_l, \mathbf{e}_k), (r_{use}, r_{select}, r_{refer\ to}, r_{contain}, r_{in}, r_{teach}, r_{offer}), (\mathbf{a}_u, \mathbf{a}_r, \mathbf{a}_t, \mathbf{a}_s)\}$, where bold letters represent sets.

5.3 FEATURE EXTRACTION

This section presents the second module of the RFKG, how to use feature extraction methods to learn different types of information in the formed KG to get features (usually in the form of feature vectors) of users and resources?

The input to this module is a formed KG from the previous module and the output is the final feature vectors of users and resources. Then, the output of this module will be used as the input of the next module (recommendation module), so the quality of the obtained features closely related to the performance of RS.

A formed KG of an e-learning platform mainly contains the semantic descriptions and values of the resources and users as well as the latent connections between multiple entities. We categorize the data contained in the formed KG into two types: *i)* the descriptions and values of resource and user are texts; *ii)* the connections between entities are represented as the graph structure. We apply two different feature extraction methods to extract the features of user and resource from the two types of information mentioned above, and then concatenate them together to form the final user and resource feature vectors, respectively.

5.3.1 *Textual attribute embedding*

The function of textual attribute embedding is to transfer textual attributes of formed KG into low-dimensional feature vectors.

The challenge is to determine the embedding algorithm. Comparing to traditional Nature Language Processing (NLP) methods, methods based on deep learning are able to obtain information more comprehensively, learn feature representations from nearly unprocessed original data and untapped data [Liang et al., 2017], such as ELMo and BERT, move beyond global word representations like Word2Vec and achieve groundbreaking performance on a wide range of natural language processing tasks [Liu et al., 2020]. However, the most advanced algorithms are not necessarily the most practical. First of all, if there are pre-trained large-scale language models, such as GPT and BERT, they can be used in recommendation of e-learning platforms with certain limitations (e.g., usage right and privacy issue). Otherwise, if these models are to be trained, a large amount of data is required. In formal e-learning platforms, such as the platforms

of MOOCs, it is not possible to train large-scale language model perfectly for textual attribute embedding when faced with data like course introductions (a platform typically contains between a few hundred and a few thousand courses). According to statistics collected from their websites, Coursera has around 7,000 courses and edX has around 3,000 courses. These amounts of data are far from sufficient for training large-scale language models to learn the features from course synopses (resource attributes).

In our formed KG, the textual attributes that need to be processed is mainly user models and resource synopses. These textual attributes contain a large number of words (e.g., age, gender, degree, format preference, and programming competency) from user models and some sentences or paragraphs from resource synopses. Classical NLP algorithms, such as Word2Vec (skip-grams and CBoW) belongs to contextual embedding methods, they need to use contextual information to learn the latent information, which do not perform well when facing with single word from user models.

To cope with these textual attributes of varying length, we propose to use Bag-of-Words (BoW) [Zhang et al., 2010] combined with autoencoder [Makhzani et al., 2015]. The BoW model a popular method for preprocessing, it pre-processes well both for single words and long texts, and auto-encoder is responsible for compressing the pre-processed features to a specific length. The architecture of textual attribute embedding is shown in Figure 5.9.

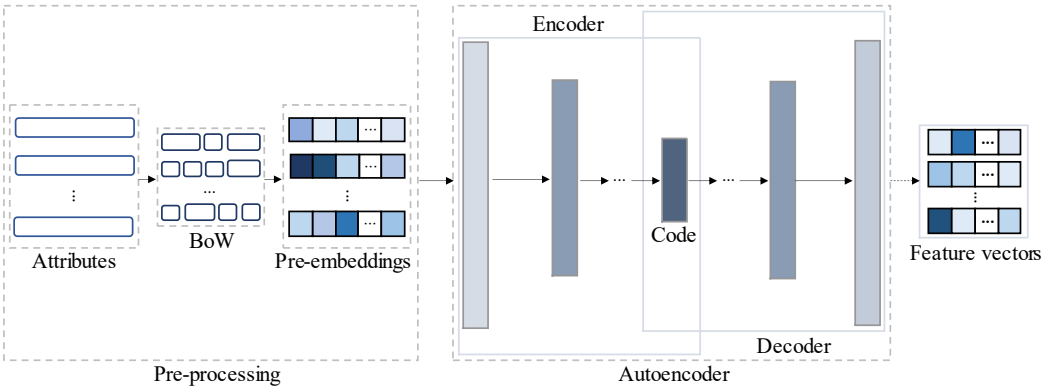


Figure 5.9: Architecture of textual attribute embedding.

5.3.1.1 *Pre-processing*

Pre-processing essentially uses BoW model to transform user models and resource synopses into pre-embeddings, which will be used as the input of autoencoder.

Using BoW model for text consists of three main steps: *segmentation* cuts the text into sequences of words; *dictionary building* counts the non-repeated words occurring in all documents and form a dictionary; *vectorization* represents each document as a word frequency vector, each element of the vector corresponds to a word in the dictionary, and its value is the number of occurrences of the word in the document. In addition to the above three steps, we can also eliminate meaningless symbols and stop words, and customize the length of the dictionary by cutting off part of the first words.

Note that, user models contain discrete text. If a user model is saved as text, it can be represented as: {*U_7001215,26,male,engineer,student,mechanics, video,French,advanced,begineert,begineer*}, corresponding to Table 5.1. Thus, compared to resource synopsis, it does not require the segmentation, but rather directly performs the dictionary building and vectorization. In addition, two BoW models are established for dealing user models and resource synopses, respectively.

5.3.1.2 *Autoencoder*

Autoencoder is an unsupervised algorithm based on low-level fully connected neural network. An autoencoder has two main parts, an encoder that maps the input into the code, a decoder that maps the code to a reconstruction of the input, and the training objective is to minimize the difference between the input and the output. The trained autoencoder model compress the input into low-dimensional vector. Encoder encodes the input x to obtain new feature $x\prime$, the encoder can be expressed as: $x\prime = f(Wx + b)$; decoder uses the feature $x\prime$ to reconstruct $\hat{x}$ that is closest to the original input x; the reconstruction error is formed between the original input x and the reconstructed $\hat{x}$, and the autoencoder learns to minimize the reconstruction error.

Let us take the process of resource attributes (synopses) as an example, the overall algorithm of textual attribute embedding is defined as:

After textual attribute embedding, the attribute feature vectors of users and resources can be obtained, denote as $\left\{\mathbf{v}_{a_u} = v_{a_{u_1}}, v_{a_{u_2}}, \cdots, v_{a_{u_{|a_u|}}}\right\}$ and $\left\{\mathbf{v}_{a_r} = v_{a_{r_1}}, v_{a_{r_2}}, \cdots, v_{a_{r_{|a_r|}}}\right\}$, respectively.

5.3.2 *Graphic structure embedding*

The function of graphic structure embedding is to learn the structure information of formed KG, capture features of entities and relations, and preserve them as low-dimensional feature vectors.

In a formed KG, a direct link between two entities indicates that there is some kind of interactive information between them. For example, in the

Algorithm 1 Textual attribute embedding

Input: All resource synopses $\{\mathbf{a}_r = a_{r_1}, a_{r_2}, \cdots, a_{r_{|\mathbf{a}_r|}}\}$

Pre-processing via BoW

$W^d \leftarrow$ Segmentation and dictionary building

$\hat{\mathbf{v}}_r \leftarrow$ Vectorization

Autoencoder

$(\theta, \phi) \leftarrow$ Initialize parameter

repeat

  $M \leftarrow$ Mini batch size

  $L = \sum_{i=1}^{|\mathbf{a}_r|/M} \left\| \hat{v}_{r_i} - g_\theta(f_\phi(\hat{v}_{r_i})) \right\|$

  $(\theta, \phi) \leftarrow$ Update parameters (e.g., SGD or Adagrad)

until Convergence of (θ, ϕ)

Input $\hat{\mathbf{v}}_r$ into convergent model to get middle code $\mathbf{v}_r$

Output: Vectors of resource attributes $\mathbf{v}_r$

domain of MOOCs, teacher t teaches a course c, which is offered by university s. The explicit interactive information can be represented as: *teacher* $t \xrightarrow{\text{teach}}$ *course c*, *university* $s \xrightarrow{\text{offer}}$ *course c*, and *teacher* $t \xrightarrow{\text{in}}$ *university s*. In addition to these visible connections, there are also a large number of invisible (i.e., latent) connections between entities in the KG. These latent connections cannot be described visually, but they can be inferred from explicit relations, and they usually contain valuable information. For example, if a user (entity) has enrolled several courses instructed by (relation) the same teacher (entity), we can infer that there is latent connection between the user and the teacher, and it provides a critical influence when the user makes decisions even though there is no direct link between them.

In order to extract the latent connections and infer valuable information from them, KGE is used to transform this information into a format usable by RSs. As mentioned in Section 3.3, KGE learns the KG and represent all the entities and relations into continuous vector spaces (can be uniform space or different spaces depending on the algorithms). Many KGE algorithms have been proposed, and thanks to the contribution of [Wang et al., 2017], we gain a comprehensive understanding of the performance between different algorithms. We summarize the conclusions:

- Models that represent entities and relations as vectors are more efficient;
- Models that represent relations as matrices or tensors usually have higher complexity in both space and time;
- Models based on neural network generally have higher complexity in time, if not in space, since matrix or even tensor computations are often required in these models;

- Expressive models do not necessarily have better performance, the reason could be that they often require a large number of parameters and tend to overfit on small-sized and medium-sized datasets.

After analyzing the conclusions of [Wang et al., 2017], TransD [Ji et al., 2015] seems to be a good choice, which is an improved translational distance model, representing entities and relations as vectors, has relatively fewer parameters, lower complexity and also the ability to cope with one-to-many and many-to-many cases. In TransD, the relations between entities are described as triplets, each entity or relation has two vectors, one represents its meaning, another is used to construct mapping matrices that project entities from entity space to relation space. The function of TransD is shown in Figure 5.10.

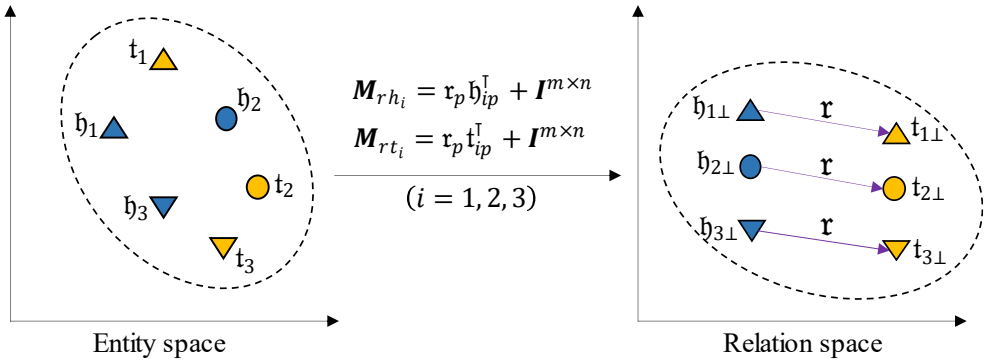


Figure 5.10: Illustration of TransD [Ji et al., 2015].

Each shape represents an entity pair appearing in a triplet of relation r. M_{rh} and M_{rt} are mapping matrices of h and t, respectively. $h_{ip}, t_{ip} (i = 1, 2, 3)$, and r_p are projection vectors. $h_{i\perp}$ and $t_{i\perp} (i = 1, 2, 3)$ are projected vectors of entities, and they conform to $h_{i\perp} + r \approx t_{i\perp} (i = 1, 2, 3)$. For a triplet $\langle h, r, t \rangle$, randomly replace the head or tail entities with other entities to form negative triple, $(h', r, t) \in \triangle'$ or $(h, r, t') \in \triangle'$, and the projected triplet is $(h_\perp, r, t_\perp)$. The score function is:

$$s = - \parallel h_\perp + r - t_\perp \parallel_2^2 \tag{5.1}$$

And the loss function is:

$$L = \sum_{\xi \in \triangle} \sum_{\xi' \in \triangle'} [\gamma + s(\xi) - s(\xi')]_+ \tag{5.2}$$

where $\triangle$ and $\triangle'$ are the sets of positive and negative triples, ξ and ξ' are golden and negative triples, $[x]_+ \triangleq max(0, x)$, and γ is the margin separating positive triplets and negative triplets.

The graphic structure embedding algorithm for our case can be described as:

Algorithm 2 Graphic structure embedding

Input: A formed KG G
1: $\langle h, r, t \rangle \leftarrow$ Export triplets from G
2: $\mathbf{v}\prime_h, \mathbf{v}\prime_r, \mathbf{v}\prime_t \leftarrow$ Embedding initialization
3: **repeat**
4: **repeat**
5: Each triplet $\xi = \langle h, r, t \rangle$, randomly replace its h or t, repeat n times to form negative triplets ξ'
6: **until** Get positive triplet set $\triangle$ and negative triplet set $\triangle'$
7: $\xi^M \leftarrow$ Mini batch
8: $L = \sum_{\xi^M \in \triangle} \sum_{\xi'^M \in \triangle'} [\gamma + s\left(\xi^M\right) - s\left(\xi'^M\right)]_+$ (equation 5.1 and 5.2)
9: $\mathbf{v}\prime_h, \mathbf{v}\prime_r, \mathbf{v}\prime_t \leftarrow$ Update embeddings
10: **until** Convergence of $\mathbf{v}_h, \mathbf{v}_r, \mathbf{v}_t$
11: Extract user and resource entity vectors from $\mathbf{v}_h, \mathbf{v}_r, \mathbf{v}_t$
Output: User entity vectors $\mathbf{v}_u$ and resource entity vectors $\mathbf{v}_r$

After graphic structure embedding, the graph feature vectors of user and resource can be obtained, denote as $\left\{\mathbf{v}_{g_u} = v_{g_{u_1}}, v_{g_{u_2}}, \cdots, v_{g_{u_{|g_u|}}}\right\}$ and $\left\{\mathbf{v}_{g_r} = v_{g_{r_1}}, v_{g_{r_2}}, \cdots, v_{g_{r_{|g_r|}}}\right\}$, respectively.

5.4 RECOMMENDATION

In this section, we design recommendation module. The two sources of feature vectors of users and resources obtained from the previous feature extraction module are concatenated respectively. The concatenated final user feature vectors and resource vectors are denoted as $\mathbf{v}_u = \left\{v_{u_1}, v_{u_2}, \cdots, v_{u_{|v_u|}}\right\}$ and $\mathbf{v}_r = \left\{v_{r_1}, v_{r_2}, \cdots, v_{r_{|v_r|}}\right\}$.

This concatenation design has been widely adopted in multimodal deep learning work [Srivastava et al., 2012] [Zhang et al., 2014] [He et al., 2017]. However, a simple vector concatenation does not account for any interactions between user and item latent features, which is insufficient for modeling the intricate effect. To address this issue, we propose to add hidden layers on the concatenated vector, using a standard MLP to learn the interaction between user and item latent features. In this case, we can endow the model a large level of flexibility and non-linearity to learn the interactions between users and resources, rather than the way that uses only a fixed element-wise product on them, such as GMF [He et al., 2017]. The MLP model is defined as:

$$
\begin{aligned}
z_1 &= \phi_1(v_{u_i}, \overline{\mathbf{v}}_r) = \begin{bmatrix} v_{u_i} \\ \overline{\mathbf{v}}_r \end{bmatrix}, \\
\phi_2(z_1) &= a_2(\mathbf{W}_2^T z_1 + \mathbf{b}_2), \\
&\dots \\
\phi_2(z_{L-1}) &= a_L(\mathbf{W}_L^T z_{L-1} + \mathbf{b}_L), \\
\hat{y}_{u_i} &= \sigma(h^T \phi_L(z_{L-1}))
\end{aligned}
\tag{5.3}
$$

where v_{u_i}, $\overline{\mathbf{v}}_r$, $\mathbf{W}_i$, $\mathbf{b}_i$, and a_i denote the i-th user, integral resource feature vector (i.e., concatenate all the resource feature vectors in a fixed order), weight matrix, bias vector, and activation function for the i-th layer's perceptron, respectively.

The architecture of the recommendation module is shown in Figure 5.11.

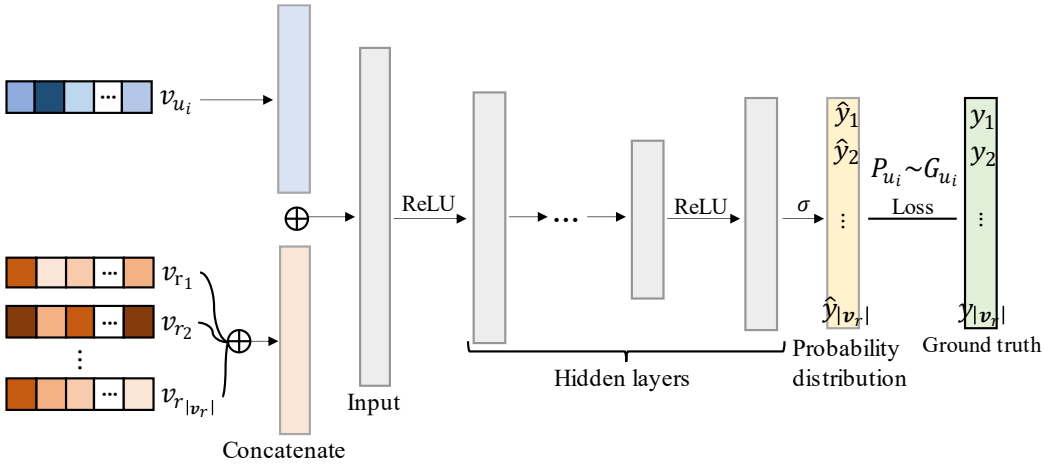


Figure 5.11: Architecture of recommendation module.

We first concatenate all the resource feature vectors in a fixed order to form the integral resource feature vector, $\overline{\mathbf{v}}_r$. The single input of the MLP is the concatenation of single user feature vector v_{u_i} and the integral resource feature vector $\overline{\mathbf{v}}_r$. We assign label (i.e., ground truth) to each input, all the resources are sorted in fixed order and the length of single label (the single label is a list) is equal to the number of the sorted resource ($|\mathbf{v}_r|$); if the user selected the resource, the value of this position in the label is set to 1, otherwise the value is set to 0. Through hidden layers, the output layer gives a probability distribution (i.e., score list), each value is between 0 and 1, represents the probability that user will select the resource in the sorted resource list.

When we perform top-K recommendation, the resources corresponding to the positions with top-K values of the probability distribution will be presented to the input user as a recommended list. During model training,

the MLP model parameters are updated by reducing the gap between the real label and predicted label. The comparation between ground truth and top-K recomandation is shown in Figure 5.12.

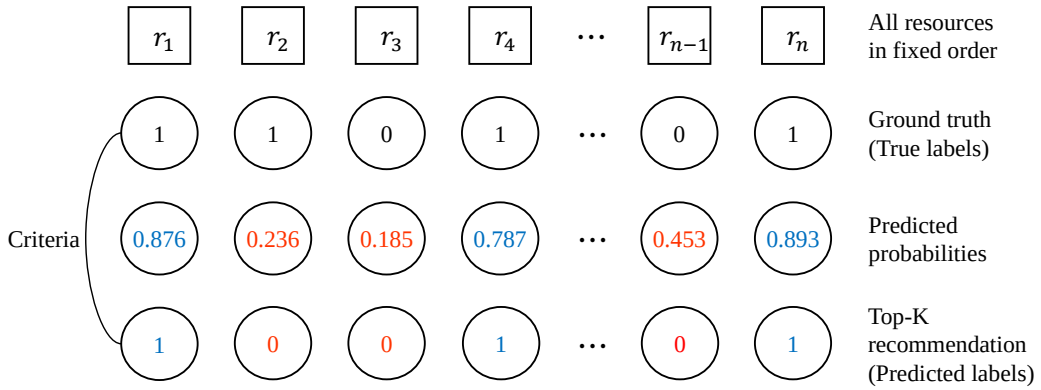


Figure 5.12: Comparation between ground truth and top-K recomandation.

5.5 CHAPTER SUMMARY

In this chapter, we addressed the research context and answered the research questions.

We want to build a RS in the e-learning platform that can make up for the flaw of conventional approaches due to most of them cannot capture the latent connections between entities in the platforms. We hope to use the powerful information representation and reasoning capabilities of KG to obtain this information that is missed by conventional approaches and use it in our RS. Therefore, we propose a recommendation framework for recommending pedagogical resources to e-learning platform users. It consists of three modules: *graph construction*, *feature extraction*, and *recommendation*. Each module is responsible for its own function and the output of the previous module is used as the input of the subsequent module to finally complete the recommendation task.

To conclude, in order to fully capture the information of e-learning platform to build RS that can assist users to select appropriate pedagogical resources, we build a lightweight KG to characterize the content of the platform. The KG contains representative entities along with their attributes, and the relations between them. In addition to the explicit information (e.g., user profile, resource synopsis, and user-resource interaction), it captures the latent connections between entities, which can provide crucial information support to the RS. At the same time, we use two feature extraction methods to obtain the features of users and resources from the

formed KG, and input them into the neural network together with the information related to user-resource interaction. The trained neural network has the ability to predict the relevance scores between unknown users and items.

Chapter 6

Experiments

In this chapter, we experimentally evaluate the performance of the proposed RFKG, and we plan to compare it with a series of baseline methods.

6.1 Datasets

To evaluate the performance of the proposed RFKG, we conduct experiments on two collected datasets, MOOCCube [Yu et al., 2020] and Open University Learning Analytics Dataset (OULAD) [Kuzilek et al., 2017]. The first dataset comes from the real-world MOOC platform *XuetangX*[1] and the second dataset comes from the distance learning platform *Open University (OU*[2]*)*.

XuetangX is one of the largest MOOC platforms in China, it offers online courses in multiple disciplines, as well as certificate and degree programs. By the end of 2022, *XuetangX* has offered nearly 6,000 high-quality courses from top universities, covering 13 disciplines. Users in *XuetangX* can choose the learning mode: Instructor-paced Mode (IPM) and Self-paced Mode (SPM). IPM follows the same course schedule as conventional classrooms, while in SPM, users could have more flexible schedule to study online by themselves. Usually an IPM course spans over 16 weeks in *XuetangX*, while a SPM course spans a longer period. Each user can enroll one or more courses. When one is studying a course, the system records multiple types of activities: video watching (watch, stop, and jump), forum discussion (ask and reply questions), assignment completion (with correct/incorrect answers and reset), and Web page clicking (click and close a Web page) [Feng et al., 2019].

OU is one of the largest distance learning universities worldwide. At present, around 170,000 students are registered in different programs on *OU*. Teaching materials and other contents are delivered to students via the Virtual Learning Environment (VLE). Students' interactions with the educational materials are recorded and stored in university data warehouse. At *OU*, courses are called modules. Modules can be presented multiple times during the year. To distinguish between different presentations of a

[1] https://www.xuetangx.com/
[2] https://www.open.ac.uk/

module, each presentation is named by the year and month it starts. For example, presentations starting in January ends with A, in February with B and so on; so that '2013J' means that the presentation started in October 2013. *OU* offers several hundred modules. Each of them can be studied as a stand-alone course or as part of a university program. No previous qualifications are required. Students in a module-presentation are organized into study groups of approximately 20 people. Each group has an assigned tutor, who guides and supports students throughout the module presentation [Kuzilek et al., 2017].

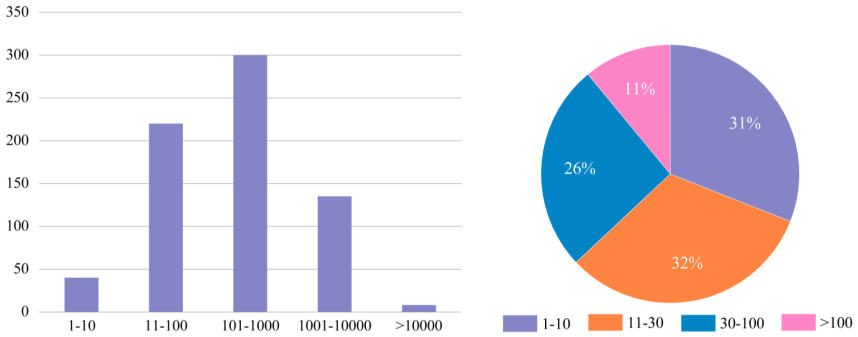
6.1.1 *MOOCCube*

MOOCCube is an open data repository for natural language processing, knowledge graphs, data mining and other researchers who are interested in MOOCs. Table 6.1 lists the statistics of MOOCCube dataset [Yu et al., 2020].

Table 6.1: Statistics of MOOCCube.

Course	Video	Concept	Prerequisite	Taxonomy	User	Enrollment	Video watching
706	38,181	106,056	17,686	3,152	199,199	682,753	4,874,298

It contains over 700 MOOC courses, 100,000 knowledge concepts, 1,99,000 users, and 8 million user behaviors with resources. Courses are the foundation of MOOCs and mainly consist of pre-recorded videos. Specifically, a course contains such as video list, course synopsis, prerequisite, and taxonomy, attached as attributes. Concepts refer to the knowledge concepts taught in the courses. The enrollment records and video watching logs are collected from 2017 to 2019, as shown in Figure 6.1.



(a) Distribution of course enrollment with users (b) Distribution of users with video watching

Figure 6.1: Distributions of (a) course enrollment with users and (b) users with video watching records of MOOCCube [Yu et al., 2020].

Figure 6.1(a) shows the course distribution of enrolled users, which substantially fits a normal distribution. Despite a few courses with rare users, 451 courses are enrolled by over 100 users. Figure 6.1(b) presents a user view of the data, indicating more than 70% of users possess over ten videos watching records. Such a wealth of data enables MOOCCube to support multiple tasks such as video navigation [Liang et al., 2015], course recommendation [Zhang et al., 2019a], concept mining [Yu et al., 2019], and dropout prediction [Feng et al., 2019].

6.1.2 *OULAD*

OULAD is a subset of the *OU* student data from 2013 and 2014. It contains the information about 7 modules (the course of *OU* is called module), 32,593 students, their assessment results, and logs of their interactions with the VLE represented by daily summaries of student clicks (10,655,280 entries) [Kuzilek et al., 2017]. Figure 6.2 is the overall structure of OULAD.

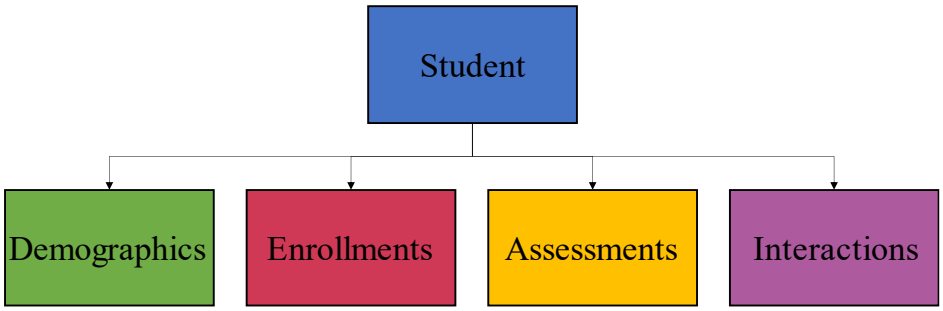


Figure 6.2: Overall structure of OULAD.

Students are linked with the information about their demographics, enrollments, assessments, and interactions for modules. For each student-module-presentation triplet, OULAD contains the results of students' assessments and the logs of interactions between students and VLE. In general, it distinguish three different data types:

- Basic information (demographic) represents the domain independent information about students;
- Activity (enrollment and interaction) is the log of student activities in the VLE;
- Performance (assessment) reflects results and achievements of students during their studies at *OU*.

Table 6.2: Main elements of the student demographic of OULAD.

Element	Description
Student id	unique student identification number;
Gender	student gender;
Region	geographic region, where the student lived while taking the module-presentation;
Highest education	highest student education level on entry to the module-presentation;
IMD band	IMD band of the place where the student lived during the module-presentation;
Age band	band of student age;
Studied credits	the total number of credits for the modules that the student is currently studying;
Disability	indicator of whether the student has declared disabilities;
Code module	module identification code which the student is registered;
Code presentation	presentation identification code which the student is registered on the module;
Final result	student final result in the module-presentation.

The dataset may be used in various scenarios, it enables evaluation of predictive models for predicting student assessment results and final module results, and recommending modules to students. It also enables researchers to study the structure of module design from the learning perspective that can be used to measure the influence of VLE on the learning outcomes and adjust the module structure to provide better VLE [Kuzilek et al., 2017].

6.2 GRAPH CONSTRUCTION

In this section, we build KGs for the above two datasets respectively. As we presented in the section 5.2, the graph construction consists of two procedures: *terminology* defines a conceptual graph model involving classes as a domain vocabulary; *assertion* imports facts (i.e., instances) into KG according to the conceptual graph model.

Note that the RFKG handles different e-learning platforms by adjusting the conceptual graph model (adjusting the classes of entities, relations, and attributes). It means that the RFKG can be applied to different e-learning platforms. Through this, we first define two conceptual graph models for MOOCCube and OULAD, then construct two KGs for them respectively.

6.2.1 *Graph construction for MOOCCube*

We extract necessary entity classes, relation classes, and attribute classes from the MOOCCube, and adjust the predefined conceptual graph model of section 5.2.1.3.

In *XuetangX*, resources are courses and learning materials mainly refer to videos. We replace the entity class *learning material* with *video* and replace the entity class *resource* with *course*. Thus, the relation class $user \xrightarrow{use} learning\ material$ is replaced with $user \xrightarrow{watch} video$, the relation class $user \xrightarrow{select} resource$ is replaced with $user \xrightarrow{enroll} course$. The modified conceptual graph model (compare to the defined model of section 5.2.1.3) for MOOCCube is shown in Figure 6.3.

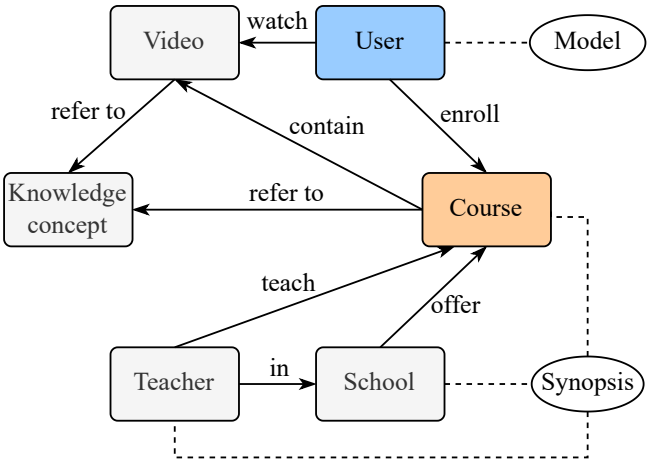


Figure 6.3: Conceptual graph model of MOOCCube.

After settling the conceptual graph model, we acquire the instances from MOOCCube for each entity class. The statistics (i.e., number of instances) for the entity classes are shown in Table 6.3.

Table 6.3: Extracted entities from MOOCCube.

User	Course	Teacher	School	Video	Knowledge concept
199,199	706	1,812	208	38,181	114,563

Before constructing KG, we optimize the extracted entities. There are a large number of users with a small number of activities, we selected the top 2,000 users in terms of the number of courses enrolled by the user, along with the 5,000 most watched videos and the 5,000 most frequently used knowledge concepts. The number of courses, teachers, and schools remain

the same. After optimization, the total number of interactions (i.e., triplets) between the remained entities is 578,751. The distribution of the number of courses enrolled by users is shown in Figure 6.4, with the average number of courses enrolled by an individual user being about 23.

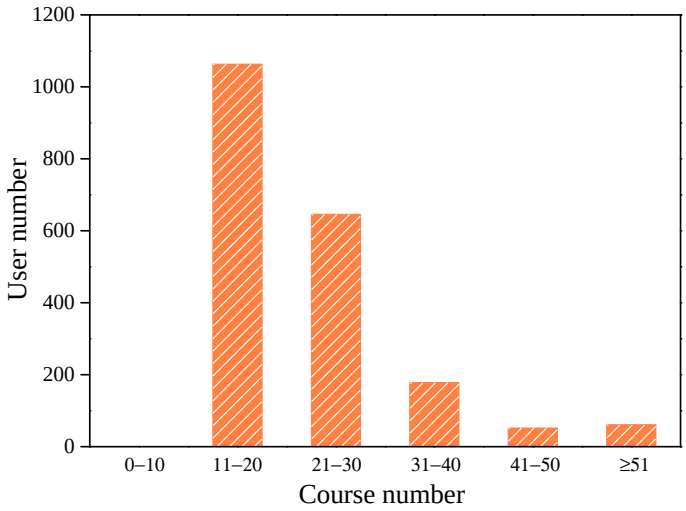


Figure 6.4: Distribution of the number of courses enrolled by users in MOOCCube.

We populate the instances into a KG database, preserve all the information in a form of KG. In this experiment, the graph database is Neo4j. Neo4j is a graph database management system, the data elements Neo4j stores are nodes, edges connecting them, and attributes of nodes and edges. We use python and the API provided by Neo4j for bulk import. The following Figure 6.5 is an illustration of the Neo4j for MOOCCube.

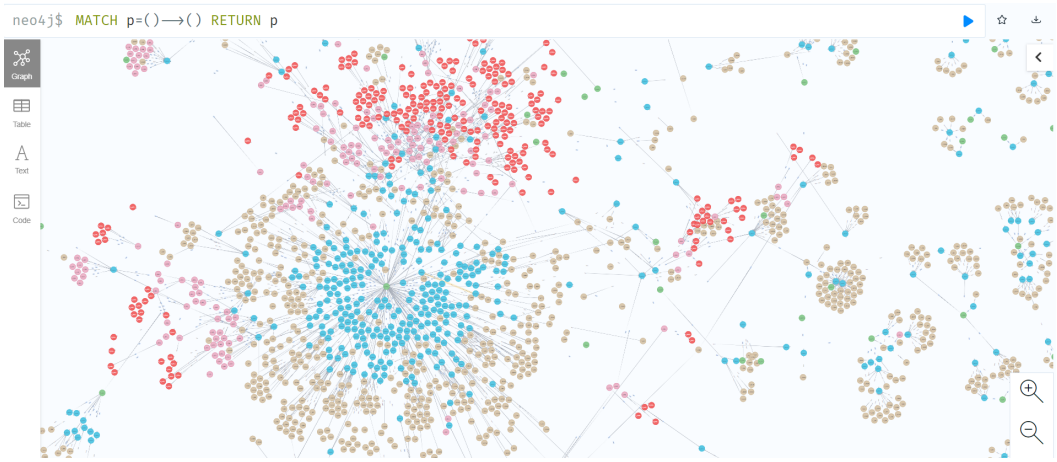


Figure 6.5: Screenshot from Neo4j running for MOOCCube. Circles of different colors represent different instances of the 6 entity classes, which are linked by 7 specific relations.

6.2.2 *Graph construction for OULAD*

Similar to section 6.2.1, we extract necessary classes and adjust the predefined conceptual graph model.

Since the the resources are called modules on *OU*, instead of courses, and modules can be presented multiple times in different semesters. To put it simply, a resource on *OU* is a combination of a module and its presentation, i.e., module-presentation. Learning materials are included in the module-presentations. In addition, OULAD does not contain information about teacher and school, but has more information about assessment. We replace the entity class *user* with *student*, replace the entity class *resource* with *module*, remove the entity class *teacher* and *school*, and add entity class *assessment* and *module-presentation*. Based on the entity classes, the relation classes can be determined: *student* $\xrightarrow{\text{use}}$ *learning material*, *student* $\xrightarrow{\text{enroll}}$ *module-presentation*, *student* $\xrightarrow{\text{has}}$ *assessment*, *learning material* $\xrightarrow{\text{in}}$ *module-presentation*, and *module-presentation* $\xrightarrow{\text{has}}$ *assessment*. The modified conceptual graph model (compare to the defined model of section 5.2.1.3) for OULAD is shown in Figure 6.6.

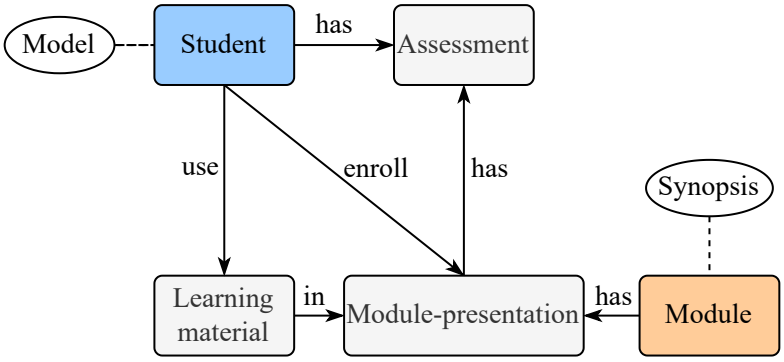


Figure 6.6: Conceptual graph model of OULAD.

After settling the conceptual graph model, we acquire the instances from OULAD for each entity class. The statistics (i.e., number of instances) for the entity classes are shown in Table 6.4.

Table 6.4: Extracted entities form OULAD.

Student	Module	Learning material	Module-presentation	Assessment
28,785	7	6,364	22	206

We also optimize the extracted entities before constructing KG. We selected the top 25,000 users in terms of the number of module-presentations enrolled by the user, along with the 6,000 most used learning materials. The number of modules, module-presentations, and assessments remain the same. After optimization, the total number of interactions (i.e., triplets) between the remained entities is 185,595. The distribution of the number of modules enrolled by students is shown in Figure 6.7, with the average number of courses enrolled by an individual student being about 1.

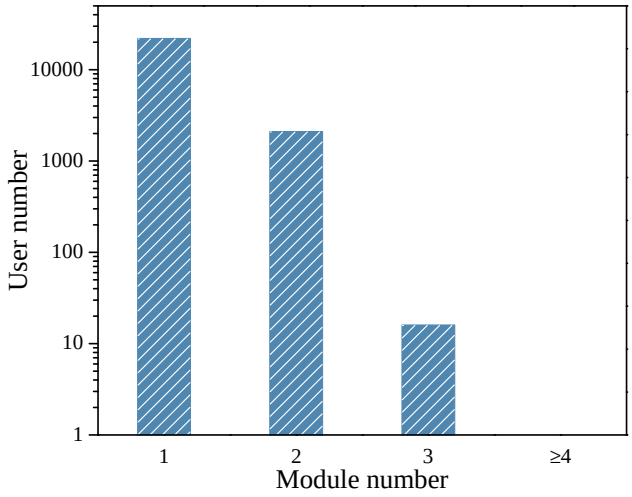


Figure 6.7: Distribution of the number of modules enrolled by students in OULAD.

We populate the instances into a KG database (also Neo4j), preserve all the information in a form of KG. The following Figure 6.8 is an illustration of the Neo4j for OULAD. Three parameters for display remain the same as section 6.2.1. Note that, after finishing the graph construction module,

Figure 6.8: Screenshot from Neo4j running for OULAD. Circles of different colors represent different instances of the 5 entity classes, which are linked by 4 specific relations.

two formed KGs (MOOCCube KG and OULAD KG) are constructed. In the next sections, we perform the feature extraction module and recommendation module.

6.3 FEATURE EXTRACTION

The process of experiments after getting constructed KGs contains three parts: *1)* BoW&autoencoder for textual attribute embedding; *2)* TransD for graphic structure embedding, and *3)* MLP for course recommendation. The inputs of the part *1)* and part *2)* are textual attributes and interaction triplets, which are derived from the constructed KGs in conformity with requirements. The input to the *3)* part is the concatenation of the outputs from the previous two parts. The process is shown in Figure 6.9.

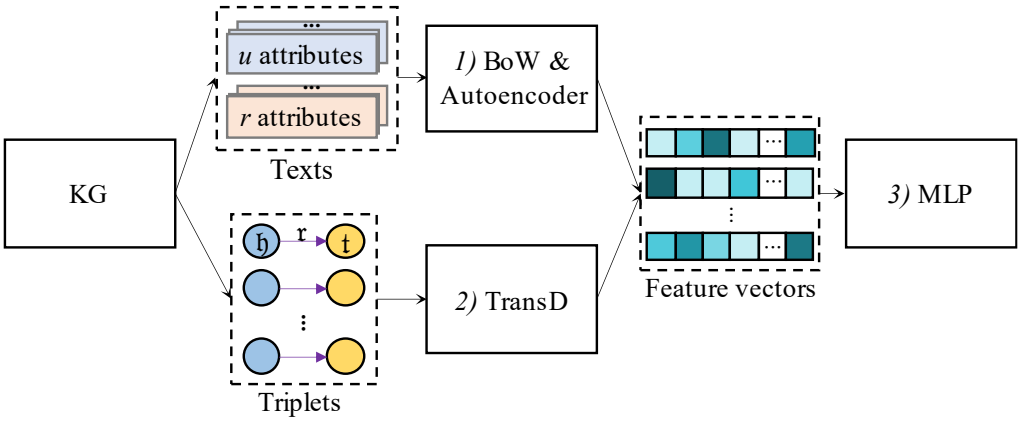


Figure 6.9: Process of the experiments after constructing KGs.

6.3.1 *Attribute embedding*

The input to attribute embedding (BoW & autoencoder) is the textual attributes of users and resources, and the output is dimension-specific feature vectors of them.

For the MOOCCube, the user model contains only *student id*, *student name*, *course order* selected by the student, and *enroll time* for these courses. There is no useful textual information that can characterize the student (because the information of *course order* and *enroll time* belong to parts of graphic structure). Instead, the course contains a synopsis paragraph which contains valuable information about the course, and there are 706 synopses (in Chinese). We perform BoW & autoencoder for the course synopses. BoW has steps including word segmentation, removal of the stop words, and construction of the word bag (the length of the bag is set to 10,000). It converts all the synopses into sparse feature vectors, and these

vectors are fed into an autoencoder network. The autoencoder network consists of an input layer, 7 hidden layers, and an output layer. All the layers are fully connected and the number of neurons in the middle layer is 8. The autoencoder outputs 706 attribute feature vectors of the courses, denote as $\left\{\mathbf{v}_{a_c} = v_{a_{c_1}}, v_{a_{c_2}}, \cdots, v_{a_{c_{706}}}\right\}$.

For the OULAD, the student model contains a lot of textual elements, and there are 25,000 entries (in English) in total. On the contrary, the module contains only *module code*, *presentation code*, and *presentation length*, which does not contain useful textual information that can characterize the module. We perform BoW & autoencoder for all student models. Although the elements in the student models are discrete, a single element may contain text with more than one word, so the process of BoW remains the same as above (in MOOCCube). It converts all the student models into sparse feature vectors, and transfers to autoencoder network. The autoencoder network consists of an input layer, 5 hidden layers, and an output layer. All the layers are fully connected and the number of neurons in the middle layer is 8. The autoencoder outputs 25,000 attribute feature vectors of the students, denote as $\left\{\mathbf{v}_{a_s} = v_{a_{s_1}}, v_{a_{s_2}}, \cdots, v_{a_{s_{25000}}}\right\}$.

6.3.2 *Structure embedding*

The input to structure embedding (TransD) consists of multiple triplets $\{\langle h, r, t\rangle\}$ (extracted from the formed KGs), and the output is dimension-specific feature vectors of all entities and relation. We then extract feature vectors of users and resources from the output. Note that, for distinction, we use $\mathbf{v}_{g_{u_i}}$ and $\mathbf{v}_{g_{c_i}}$ to represent user and course structure feature vector sets from MOOCCube; $\mathbf{v}_{g_{s_i}}$ and $\mathbf{v}_{g_{m_i}}$ are used to represent student and module structure feature vector sets from OULAD.

In the process of TransD, we first create negative pool with negative sampling (random replacement of head and tail in for a triplet). Then, we build TransD model with the help of Pytorch, which initializes all embeddings of entities and relations. Finally, under the constraint of the loss function (see equation 5.2), the model learns the difference between positive and negative samples and updates the embeddings. Note that some parameters of the model are: the optimizer is Adam; the learning rate is 1e-3; the batch size is 128; the epoch is 100; and the embedding dimension is set to 128.

In the actual operation of training process, we built two TransD models for the two datasets respectively, where the parameters remain the same as described above. The average epoch loss of the two TransD models during the training process is shown in Figure 6.10.

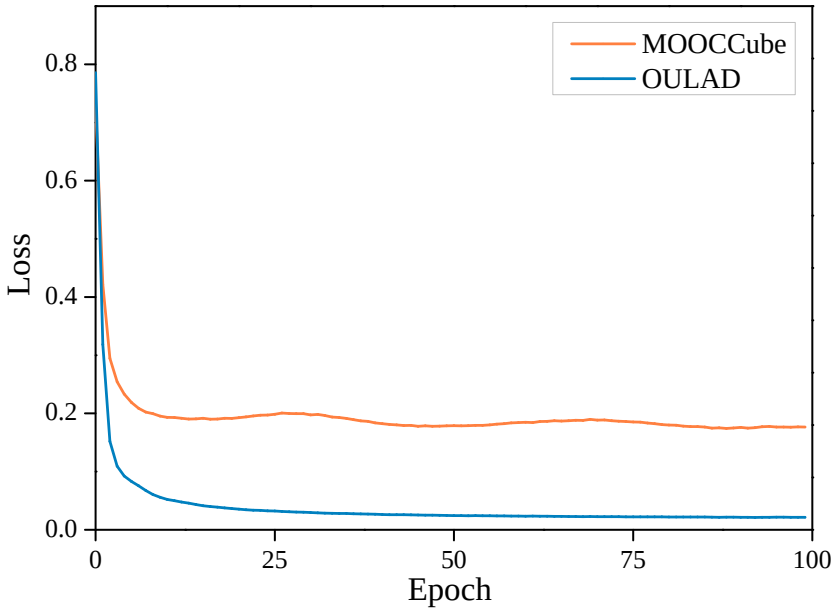


Figure 6.10: Training loss of TransD models for MOOCCube and OULAD.

After finishing the training, we extract structure feature vectors of user, courses, students, and modules from the two outputs, denote as $\left\{\mathbf{v}_{g_u} = v_{g_{u_1}}, v_{g_{u_2}}, \cdots, v_{g_{u_{2000}}}\right\}$, $\left\{\mathbf{v}_{g_c} = v_{g_{c_1}}, v_{g_{c_2}}, \cdots, v_{g_{c_{706}}}\right\}$, $\left\{\mathbf{v}_{g_s} = v_{g_{s_1}}, v_{g_{s_2}}, \cdots, v_{g_{s_{25000}}}\right\}$, and $\left\{\mathbf{v}_{g_m} = v_{g_{m_1}}, v_{g_{m_2}}, \cdots, v_{g_{m_7}}\right\}$, respectively.

6.4 RECOMMENDATION

The main component of recommendation module is a MLP model. Different datasets make it different structures (e.g., number of hidden layer, input/output size, activation function, and loss function). The single input of MLP is a combination of single user feature vector and overall resource feature vector (concatenate all resource feature vectors in a fixed order). The output is a probability distribution (in a fixed order) over all resources.

In the process of recommendation, we first concatenate the user feature vectors with the overall resource feature vector one by one and, at the same time, create labels for each user based on the resource selected by the user. Then, we cut the concatenated vectors and labels to compose the training data and testing data. Next, we build the MLP model with Pytorch and train the model with training data (the goal of training is to minimize the gap between the model outputs and the actual labels). Note that some parameters of the model are: the model has 7 fully connected feedforward layers; the optimizer is Adam; the learning rate is 1e-3; the batch size is 128; and the epoch is 100.

In the actual operation of training process, we built two MLP models for the two datasets respectively. Among them, except for the different number of neurons in each layer of the MLP models, other parameters remain the same as stated above. The training loss of MLP models for the two datasets is shown in Figure 6.11.

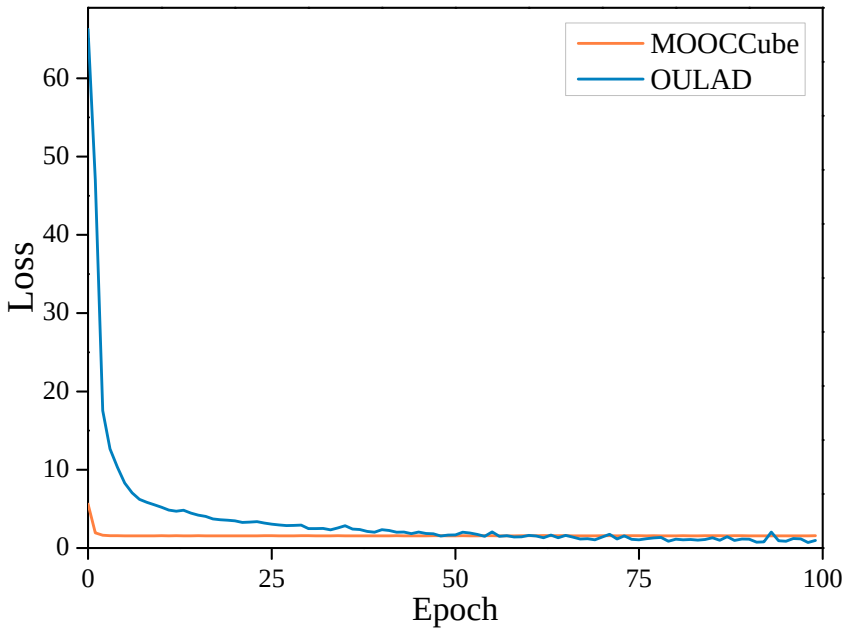


Figure 6.11: Training loss of MLP models for MOOCCube and OULAD.

In such recommendation scenario, the user feedback refers to the interactions between users and resources (i.e., whether there is an interaction between user and resource, the relevance score is 1 if there is an interaction, otherwise the relevance score is 0), which belongs to implicit feedback. Accuracy is not a suitable performance measurement for implicit feedback recommendation. For the determination of evaluation matrices, we choose two most wildly used metrics, HR (equation 4.27) and NDCG (equation 4.26), for such an implicit feedback recommendation scenario.

When evaluating the top-K RSs, we need to set the corresponding K values. In our case, we set the K to 1, 5, 10, and 15 for MOOCCube; differently, we set the K to 1 and 3 for the OULAD. The reason for such settings is that the resources of MOOCCube are 706 courses, and the number of course enrollments per user after optimization is 23; while the resources of OULAD are only 7 modules, and the number of module enrollments per student after optimization is 1. Compared with MOOCCube, the item pool is very small, only 7 items in the poll. Therefore, an excessively high K value is inappropriate for OULAD.

The performance of the proposed framework measured by HR and NDCG with different K values on the two datasets is shown in Figure 6.12.

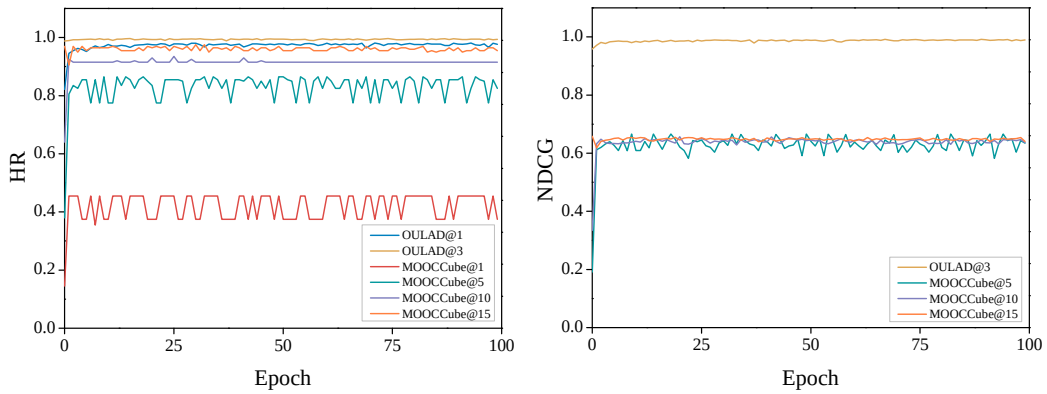


Figure 6.12: Performance of the proposed framework measured by HR and NDCG with different K values on the two datasets.

We can see from the trend of the results in the figure that our framework performs well on both datasets, but the performance on OULAD is much better ('too good') than the performance on MOOCCube. As for the reason of the overly excellent results on OULAD, we analyze that there are two possible causes: *1)* the item pool (resource pool) of OULAD is too small (only 7 modules), MLP can effectively distinguish them; *2)* 91.2% of students chose only one module (resource), the relevance between students and courses is very single, which is easy for the MLP to capture. In fact, we deem that the experiments on MOOCCube is more informative because the distribution of the number of course enrollment with users is more rational and the item pool is sufficient.

6.5 BASELINE METHODS

To compare the performance of the proposed framework with state of the art, we adopt the following baseline methods from numerous top-K recommendation approaches.

BPR [Rendle et al., 2014]: Bayesian Personalized Ranking (BPR) is a commonly used recommendation algorithm in current RSs. Different from other methods based on user rating matrices, BPR mainly uses users' implicit feedback (such as clicks, favorites, etc.) to rank items with the maximum posterior probability obtained by performing Bayesian analysis of the problem, and then generate recommendations.

MLP [He et al., 2017]: Multi-Layer Perceptron (MLP) is a kind of feedforward neural network, consisting of fully connected neurons with a nonlinear activation function, organized in at least three layers. In RSs, MLP

receives vector representations of users and items and learns the correlation between them, then recommends potentially relevant items to the target user.

FISM [Kabbur et al., 2013]: FISM a factored item similarity based method for the top-K recommendation problem. It learns the item similarity matrix as the product of two low dimensional latent factor matrices, where these matrices are learned using a structural equation modeling approach.

NAIS [He et al., 2018]: Neural Attentive Item Similarity (NAIS) is a neural network for item-based collaborative filtering recommendation. The key argument is that the historical items of a user profile do not contribute equally to predict the user's preference on an item. NAIS first revisited the FISM method from the perspective of representation learning, and then devised several attention mechanisms step by step to enhance its representation ability. At the same time, it proposed a effective variant of softmax to address the large variance issue on user behaviors.

GMF [He et al., 2017]: General Matrix Factorization (GMF) is the most popular model for recommendation and has been investigated extensively in literature, it associates each user and item with a real-valued vector of latent features, models the two-way interaction of user and item latent factors, assuming each dimension of the latent space is independent of each other and linearly combining them with the same weight, which can be deemed as a linear model of latent factors.

NeuMF [He et al., 2017]: NeuMF fuses MF and MLP under a neural collaborative filtering framework, MF that applies a linear kernel to model the latent feature interactions, and MLP that uses a non-linear kernel to learn the interaction function from data.

6.6 RESULTS AND DISCUSSION

The next comparative experiments we conducted include two aspects:

- The performance between RFKG and other baseline methods under default embedding sizes (attribute embedding size is 8 and structure embedding size is 128);
- The impact of different embedding sizes on the recommendation performance of RFKG.

First, we perform BoW & autoencoder and TransD for the two datasets with the default embedding sizes (attribute embedding size is 8 and structure embedding size is 128), and train the MLP model with their output. As concluded in section 6.4, MOOCCube is deemed to be more informative compare with OULAD. So, after completing the training process, we use MOOCCube to compare the performance of our framework and baseline methods by HR and NDCG with different K values. The results are

shown in Table 6.5, the best performance and the second best performance methods are denoted in bold and underlined fonts respectively.

Table 6.5: Performance evaluations measured by HR and NDCG with different K values for the proposed framework and baseline methods on MOOCCube.

Dataset	Metric	BPR	FISM	NAIS	MLP	NeuMF	GMF	RFKG
MOOCCube	HR@1	0.2151	0.1447	0.1514	0.2125	0.2300	0.2335	**0.4550**
	HR@5	0.4172	0.4014	0.4051	0.5200	0.5380	0.5545	**0.8750**
	NDCG@5	0.2752	0.2765	0.2786	0.3665	0.3870	0.3923	**0.6441**
	HR@10	0.5070	0.5884	0.5829	0.7200	0.7160	0.7340	**0.9250**
	NDCG@10	0.2646	0.3339	0.3352	0.4404	0.4427	0.4545	**0.6314**
	HR@15	0.5623	0.7032	0.7046	0.8050	0.8070	0.8370	**0.9800**
	NDCG@15	0.2963	0.3662	0.3683	0.4550	0.4603	0.4823	**0.6567**

Based on the results, we can draw the following conclusion: RFKG outperforms other baseline methods. We visualize the experimental results on MOOCCube, as shown in Figure 6.13.

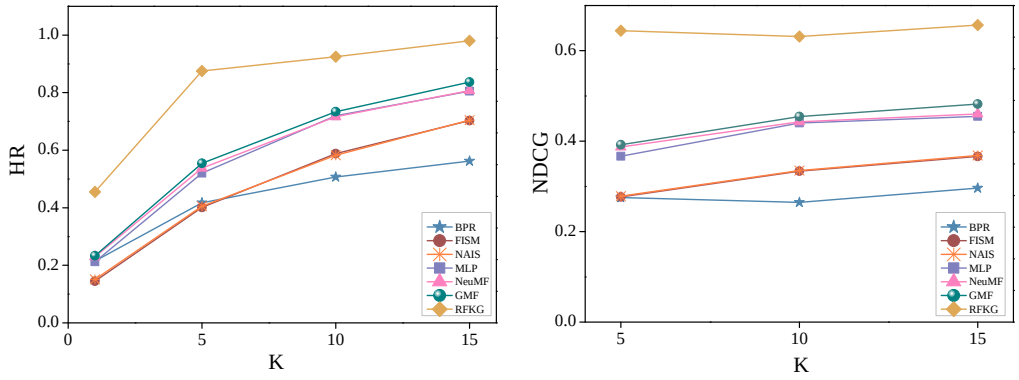


Figure 6.13: Visualization of the experimental results on MOOCCube.

We analyze the mechanism of RFKG to investigate why it performs better than the baseline methods. Among the three modules of the framework (*graph construction*, *feature extraction*, and *recommendation*), the first two modules can be regarded as a pre-training process, which obtains some of the features of the users and resources from the text (e.g., user profile and resource synopsis) and graph structure in advance (compared with the use of one hot encoding in some methods, this process can obtain the features in a more reasonable way); these obtained features, together with the user-resource interactions, are inputted into the third module (a neural network), which learns the intrinsic relevance between users and resources, and these pre-features provide a priori knowledge to the neural network, thereby optimizing the training process.

The embedding size is a factor that affect the recommendation performance. In our framework, there are two embedding methods, textual attribute embedding and graphic structure embedding, we separately study the impact of their different settings of embedding size on the final recommendation performance. When studying one embedding method, the embedding size of another one should be the default (the default sizes of attribute embedding and structure embedding are 8 and 128). We perform BoW&autoencoder and TransD for the MOOCCube with different embedding sizes (8, 16, 32, 64, and 128), and train the MLP model with their output respectively. After completing the training process, we evaluate the performance of our framework on NDCG@10. The results are shown in Figure 6.14.

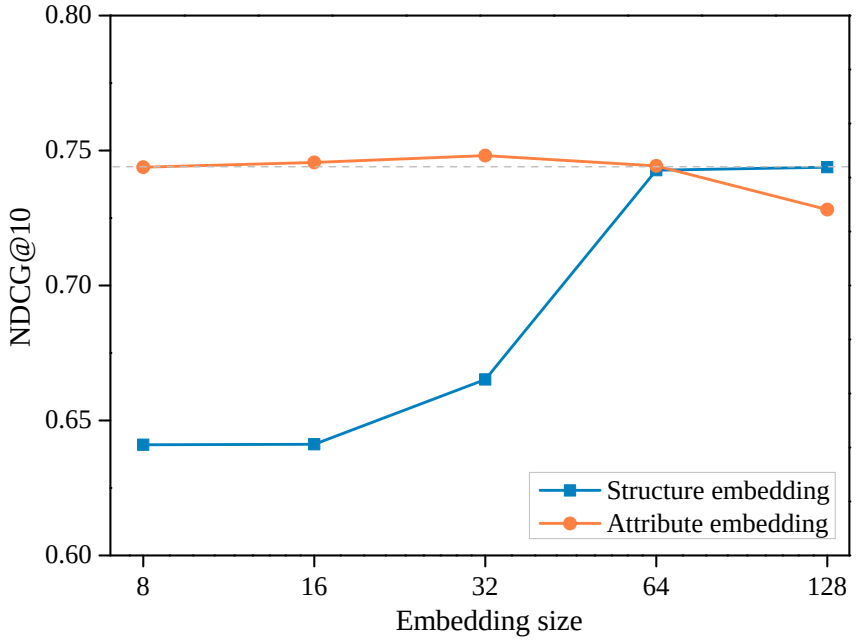


Figure 6.14: Performance (NDCG@10) comparison of different embedding (attribute embedding and structure embedding) sizes on MOOCCube.

In the above figure, for the investigation of structure embedding size, the settings (attribute embedding size, structure embedding size) of the blue line from left to right are (8, 8), (8, 16), (8, 32), (8, 64) and (8, 128), and the recommendation performance is getting better as the structure embedding size increases; for the investigation of attribute embedding size, the settings (attribute embedding size, structure embedding size) of the orange line from left to right are (8, 128), (16, 128), (32, 128), (64, 128) and (128, 128), the recommendation performance first increases and then decreases. Because the graphic structure is complex and contains more features than textual attribute, a larger capacity feature vector is required to store these features. The length of textual attributes is not too long and the conservation of features only require 32-dimensional feature vector (NDCG@10

starts to decline when attribute embedding size greater than 32). The feature vector that is too long will make the MLP network more complex and will have a negative impact on recommendation performance.

Thus, we can draw a conclusion that the recommendation performance is getting better as the embedding size increases within a certain range.

6.7 CHAPTER SUMMARY

In this chapter, our work is mainly related to the evaluation of the proposed recommendation framework. Firstly, we choose two datasets collected from real-world e-learning platforms. Secondly, in addition to conducting experiments on the framework, we also select a series of base methods for performance comparison, and the results show that our recommendation framework outperforms the other base methods on the two datasets. Finally, we also conduct a series of experiments for the embedding size of the two embedding methods (attribute embedding and structure embedding) in the framework, and we conclude that the final recommendation performance is directly proportional to the embedding size, regardless of whether it is attribute embedding or structure embedding.

Part IV

CONCLUSIONS AND PERSPECTIVES

Chapter 7

Conclusions and perspectives

7.1 conclusions

As an important branch of education, e-learning is in a phase of rapid development, more and more resources are being placed in a variety of e-learning platforms, which brings convenience to users and also generates a heavy information burden to them. RSs are applied in these platforms to cope with this problem. A good RS should consider as much information as possible that can influence the user's choice, so as to accurately predict the user's next choice. However, conventional RSs mainly focus on information directly relevant to users or resources. CF RSs make recommendations by learning user-item interactions, either through explicit (e.g., user ratings) or implicit feedback (e.g., browsing history) [Zhang et al., 2019b]. CB RSs make recommendations by comparing the representations of item content to the representations of user interests [Melville et al., 2010]. Hybrid RSs combine two or more recommendation strategies in different ways to benefit from their complementary advantages [Çano et al., 2017]. Most of the conventional RSs ignore a piece of crucial information, the latent connections between the entities in the specific domain. For example, *Qing* tries to find a course related to machine learning on Coursera; he chose the course *Machine learning* launched by *Stanford University*, instead of *Machine learning* launched by *University of Washington*; the choice is independent of the content of the two courses, only his subjective feelings (i.e., he prefers *Stanford University*); so, we can assert that there is a latent connection between him and the school, and that connection influences his selection. Therefore, research focuses on how to obtain these mentioned latent connections and transform them into information usable by RSs.

We notice that KG has strong capabilities in information representation and information reasoning [Chen et al., 2020]. Perhaps KG can help us obtain these latent connections. After our investigation, we believe that KG can not only store regular data that can be obtained by traditional approaches, but also capture the latent connections between the lurking entities of e-learning platform and make them machine-readable, which will make it possible to turn them into information usable by RSs.

Thus, this thesis address two main research questions:

1. How to characterize the content of e-learning platform by the means of KG? Can the KG be generic?

2. How to obtain features of users and resources from the constructed KG? How to learn these features with algorithm for recommendation?

In order to answer the first question, we build a lightweight KG that captures as much information (both ordinary data and latent connections) as possible that affects the user's choice and turns it into machine readable. Specifically, we follow the core steps to create a lightweight ontology represented as a KG for e-learning platform [Noy et al., 2001]. *step 1.* Determine the domain and scope of the ontology; *step 2.* Define the entity classes and their relation classes; *step 3.* Define the properties (i.e., attributes) of classes; *step 4.* Extract the instances of the defined classes and compress them into a knowledge base represented as a KG. Note that, the defined classes can be adjusted according to different e-learning platforms, which makes the KG generic.

In order to answer the second question, we first use NLP technologies to process the descriptions and values of the resources and users to get text feature vectors of users and resources. Then, we use KGE algorithm to learn the latent information from the graph structure to get structure feature vectors of users and resources. The obtained text and structure feature vectors are concatenated to form the final feature vectors of users and resources respectively, and input into a MLP network for model fitting. The trained MLP network has capability to predict the relevance between different users and resources and make resource recommendations to target users.

For evaluating the performance of the recommendation framework, we conduct a series of experiments by using datasets form real-world e-learning platforms (*XuetangX* and *Open University*), the results demonstrate that our recommendation framework outperforms methods from the state-of-the-art on the chosen datasets.

7.2 PERSPECTIVES

The perspectives can be described in two aspects: short-term work and long-term future work.

Short-term work

The short-term work is mainly about optimizing the established recommendation framework. The recommendation accuracy of downstream

RS is closely related to the obtained features from upstream feature extraction. In our framework, we introduce KG to characterize the content of e-learning platform so as to capture more sources of information that capture user and resource features. But for the KGE algorithm to learn the formed KG to obtain the feature vectors, we just chose TransD based on the results from the stat of the art [Wang et al., 2017]. We have not attempted to use other KGE algorithms to learn the formed KG, especially since more and more Graph Neural Networks (GNNs) algorithms have been proposed in the last few years. GNN is a class of artificial neural networks for processing data that can be represented as graphs [Scarselli et al., 2008] [Sanchez-Lengeling et al., 2021] [Wu et al., 2022]. The key design element of GNNs is the use of pairwise message passing, such that graph nodes iteratively update their representations by exchanging information with their neighbors. Several different GNN architectures have been proposed, which implement different mechanisms of message passing [Kipf et al., 2016] [Hamilton et al., 2017] [Veličković et al., 2017] [Bronstein et al., 2021].

Can GNN obtains higher quality features from the KG than the traditional KGE algorithm TransD? On the basis of the currently used dataset, we will conduct experiments using other latest conventional KGE algorithms besides TransD, and also GNNs (e.g., Graph Convolutional Network (GCN) [Kipf et al., 2016] and Graph Attention Network (GAN) [Veličković et al., 2017]) for the performance comparison.

Long-term future work

The long-term future work is mainly about investigating the scalability and robustness of the established recommendation framework. For the scalability, there are only 6 core entity classes defined in the KG, but in fact, other entity classes may exist in the e-learning platform, such as subject and prerequisite.

Can the addition of these entity classes effectively enhance the ability of KG to characterize the e-learning platform? We will try to add more potentially relevant entity classes to the designed conceptual graph model and rebuild the KG. Then, both feature extraction and recommendation will be performed as designed before. Finally, the performance based on the new KG will be compared with the old results.

Furthermore, although we validated the effectiveness of our recommendation framework on two datasets collected from e-learning platform *XuetangX* and *Open university*. But these two platforms are essentially both MOOC platforms, and we still need to test our framework on datasets from other non-MOOC platforms to verify its robustness.

Does our recommendation framework perform as expected on other non-MOOC platforms? We will continue to follow up the experiments,

and look forward to obtaining other datasets to verify the feasibility of our recommendation framework on other non-MOOC platforms.

Bibliography

Aczel, James C, Stephen R Peake, and P Hardy (2008). "Designing capacity-building in e-learning expertise: Challenges and strategies". In: *Computers & Education* 50.2, pp. 499–510 (cited on p. 3).

Adomavicius, Gediminas and Alexander Tuzhilin (2005). "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions". In: *IEEE transactions on knowledge and data engineering* 17.6, pp. 734–749 (cited on pp. 34, 38).

— (2010). "Context-aware recommender systems". In: *Recommender systems handbook*. Springer, pp. 217–253 (cited on p. 44).

Aggarwal, Charu C and Charu C Aggarwal (2016a). "Ensemble-based and hybrid recommender systems". In: *Recommender Systems: The Textbook*, pp. 199–224 (cited on p. 39).

— (2016b). "Evaluating recommender systems". In: *Recommender Systems: The Textbook*, pp. 225–254 (cited on p. 40).

— (2016c). "Model-based collaborative filtering". In: *Recommender systems: the textbook*, pp. 71–138 (cited on p. 32).

Aldowah, Hanan, Hosam Al-Samarraie, Ahmed Ibrahim Alzahrani, and Nasser Alalwan (2020). "Factors affecting student dropout in MOOCs: a cause and effect decision-making model". In: *Journal of Computing in Higher Education* 32, pp. 429–454 (cited on pp. 14–16).

Angles, Renzo, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan Reutter, and Domagoj Vrgoč (2017). "Foundations of modern query languages for graph databases". In: *ACM Computing Surveys (CSUR)* 50.5, pp. 1–40 (cited on pp. 21, 22).

Ansari, Asim, Skander Essegaier, and Rajeev Kohli (2000). *Internet recommendation systems* (cited on p. 39).

Balabanović, Marko and Yoav Shoham (1997). "Fab: Content-Based, Collaborative Recommendation". In: *Commun. ACM* 40.3, 66–72. ISSN: 0001-0782 (cited on pp. 38, 39).

Balazevic, Ivana, Carl Allen, and Timothy Hospedales (2019). "Multi-relational poincaré graph embeddings". In: *Advances in Neural Information Processing Systems* 32 (cited on p. 23).

Balažević, Ivana, Carl Allen, and Timothy M Hospedales (2019a). "Hypernetwork knowledge graph embeddings". In: *Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28*. Springer, pp. 553–565 (cited on p. 25).

Balažević, Ivana, Carl Allen, and Timothy M Hospedales (2019b). "Tucker: Tensor factorization for knowledge graph completion". In: *arXiv preprint arXiv:1901.09590* (cited on p. 25).

Barak, Miri, Abeer Watted, and Hossam Haick (2016). "Motivation to learn in massive open online courses: Examining aspects of language and social engagement". In: *Computers & Education* 94, pp. 49–60. ISSN: 0360-1315 (cited on p. 16).

Baturay, Meltem Huri (2015). "An overview of the world of MOOCs". In: *Procedia-Social and Behavioral Sciences* 174, pp. 427–433 (cited on pp. 14, 15).

Behera, Santosh Kumar (2013). "E-and M-Learning: A comparative study". In: *International Journal on New Trends in Education and Their Implications* 4.3, pp. 65–78 (cited on p. 3).

Bichsel, Jacqueline (2013). "The state of e-learning in higher education: An eye toward growth and increased access". In: *Educause Center for analysis and research* (cited on p. 13).

Blanco, Roi, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec (2013). "Entity recommendations in web search". In: *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II 12*. Springer, pp. 33–48 (cited on p. 20).

Blin, Françoise and Morag Munro (2008). "Why hasn't technology disrupted academics' teaching practices? Understanding resistance to change through the lens of activity theory". In: *Computers & Education* 50.2, pp. 475–490 (cited on p. 14).

Bobadilla, Jesus, Santiago Alonso, and Antonio Hernando (2020). "Deep learning architecture for collaborative filtering recommender systems". In: *Applied Sciences* 10.7, p. 2441 (cited on p. 40).

Bollacker, Kurt, Patrick Tufts, Tomi Pierce, and Robert Cook (2007). "A platform for scalable, collaborative, structured information integration". In: *Intl. Workshop on Information Integration on the Web (IIWeb'07)*, pp. 22–27 (cited on p. 20).

Bordes, Antoine, Xavier Glorot, Jason Weston, and Yoshua Bengio (2014). "A semantic matching energy function for learning with multi-relational data: Application to word-sense disambiguation". In: *Machine Learning* 94, pp. 233–259 (cited on p. 25).

Bordes, Antoine, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko (2013). "Translating embeddings for modeling multi-relational data". In: *Advances in neural information processing systems* 26 (cited on p. 23).

Breese, John S, David Heckerman, and Carl Kadie (2013). "Empirical analysis of predictive algorithms for collaborative filtering". In: *arXiv preprint arXiv:1301.7363* (cited on pp. 29, 32).

Breslow, Lori, David Pritchard, Jennifer DeBoer, Glenda Stump, Andrew Ho, and Daniel Seaton (June 2013). "Studying Learning in the Worldwide Classroom: Research into edX's First MOOC". In: *Research in Practice and Assessment* (cited on p. 16).

Bronstein, Michael M, Joan Bruna, Taco Cohen, and Petar Veličković (2021). "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges". In: *arXiv preprint arXiv:2104.13478* (cited on p. 91).

Burke, Robin (2000). "Knowledge-based recommender systems". In: *Encyclopedia of library and information systems* 69.Supplement 32, pp. 175–186 (cited on pp. 39, 44).

Çano, Erion and Maurizio Morisio (2017). "Hybrid recommender systems: A systematic literature review". In: *Intelligent data analysis* 21.6, pp. 1487–1524 (cited on p. 89).

Cao, Yixin, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua (2019). "Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences". In: *The world wide web conference*, pp. 151–161 (cited on p. 44).

Chen, Hui, Chuantao Yin, Xin Fan, Lei Qiao, Wenge Rong, and Xiong Zhang (2021). "Learning path recommendation for MOOC platforms based on a knowledge graph". In: *Knowledge Science, Engineering and Management: 14th International Conference, KSEM 2021, Tokyo, Japan, August 14–16, 2021, Proceedings, Part II 14*. Springer, pp. 600–611 (cited on p. 45).

Chen, Jing, Jun Feng, Xia Sun, Nannan Wu, Zhengzheng Yang, and Sushing Chen (2019). "MOOC dropout prediction using a hybrid algorithm based on decision tree and extreme learning machine". In: *Mathematical Problems in Engineering* 2019 (cited on p. 16).

Chen, Xiaojun, Shengbin Jia, and Yang Xiang (2020). "A review: Knowledge reasoning over knowledge graph". In: *Expert Systems with Applications* 141, p. 112948 (cited on pp. 19, 89).

Cimiano, Philipp and Heiko Paulheim (2017). "Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods". In: *Semant. Web* 8.3, 489–508 (cited on p. 19).

Costa, Carolina, Helena Alvelos, and Leonor Teixeira (2012). "The use of Moodle e-learning platform: a study in a Portuguese University". In: *Procedia Technology* 5, pp. 334–343 (cited on p. 14).

Cremonesi, Paolo, Yehuda Koren, and Roberto Turrin (2010). "Performance of recommender algorithms on top-n recommendation tasks". In: *Proceedings of the fourth ACM conference on Recommender systems*, pp. 39–46 (cited on p. 40).

Deng, Ruiqi, Pierre Benckendorff, and Deanne Gannaway (2019). "Progress and new directions for teaching and learning in MOOCs". In: *Computers & Education* 129, pp. 48–60 (cited on p. 15).

Dettmers, Tim, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel (2018). "Convolutional 2d knowledge graph embeddings". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1 (cited on p. 25).

Dong, Xin, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang (2014). "Knowledge vault: A web-scale approach to probabilistic knowledge fusion". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 601–610 (cited on p. 25).

Downes, Stephen (2005). "E-learning 2.0". In: *ELearn* 2005.10, p. 1 (cited on pp. 3, 11).

Ehrlinger, Lisa and Wolfram Wöß (2016). "Towards a definition of knowledge graphs." In: *SEMANTiCS (Posters, Demos, SuCCESS)* 48.1-4, p. 2 (cited on p. 20).

Ezaldeen, Hadi, Rachita Misra, Sukant Kishoro Bisoy, Rawaa Alatrash, and Rojalina Priyadarshini (2022). "A hybrid E-learning recommendation integrating adaptive profiling and sentiment analysis". In: *Journal of Web Semantics* 72, p. 100700 (cited on p. 4).

Färber, Michael, Frederic Bartscherer, Carsten Menne, and Achim Rettinger (2018). "Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago". In: *Semantic Web* 9.1, pp. 77–129 (cited on p. 19).

Feng, Wenzheng, Jie Tang, and Tracy Xiao Liu (2019). "Understanding dropouts in MOOCs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01, pp. 517–524 (cited on pp. 15, 69, 71).

Fernandez, Cassandra Jane, Rachana Ramesh, and Anand Shankar Raja Manivannan (2022). "Synchronous learning and asynchronous learning during COVID-19 pandemic: a case study in India". In: *Asian Association of Open Universities Journal* (cited on p. 12).

Frehywot, Seble, Yianna Vovides, Zohray Talib, Nadia Mikhail, Heather Ross, Hannah Wohltjen, Selam Bedada, Kristine Korhumel, Abdel Karim Koumare, and James Scott (2013). "E-learning in medical education in resource constrained low-and middle-income countries". In: *Human resources for health* 11, pp. 1–15 (cited on p. 3).

Froschl, Christoph (2005). "User modeling and user profiling in adaptive e-learning systems". In: *Graz, Austria: Master Thesis* (cited on p. 17).

Gong, Jibing, Shen Wang, Jinlong Wang, Wenzheng Feng, Hao Peng, Jie Tang, and Philip S Yu (2020). "Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view". In: *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp. 79–88 (cited on p. 3).

Grainger, Barney (2013). "Massive open online course (MOOC) report 2013". In: *University of London* (cited on p. 15).

Gunawardana, Asela, Guy Shani, and Sivan Yogev (2012). "Evaluating recommender systems". In: *Recommender systems handbook*. Springer, pp. 547–601 (cited on p. 40).

Guo, Qingyu, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He (2020). "A survey on knowledge graph-based recommender systems". In: *IEEE Transactions on Knowledge and Data Engineering* 34.8, pp. 3549–3568 (cited on pp. 42–44).

Hadi, Syed Munib and Phillip Gagen (Feb. 2016). "New model for measuring MOOCs completion rates". In: (cited on p. 16).

Hamilton, Will, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec (2018). "Embedding logical queries on knowledge graphs". In: *Advances in neural information processing systems* 31 (cited on p. 22).

Hamilton, Will, Zhitao Ying, and Jure Leskovec (2017). "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (cited on p. 91).

Hao, Pengyi, Yali Li, and Cong Bai (2023). "Meta-relationship for course recommendation in MOOCs". In: *Multimedia Systems* 29.1, pp. 235–246 (cited on p. 45).

Hassenburg, Amy (2009). "Distance education versus the traditional classroom". In: *Berkeley Scientific Journal* 13.1 (cited on p. 11).

He, Xiangnan, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua (2018). "Nais: Neural attentive item similarity model for recommendation". In: *IEEE Transactions on Knowledge and Data Engineering* 30.12, pp. 2354–2366 (cited on p. 82).

He, Xiangnan, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua (2017). "Neural collaborative filtering". In: *Proceedings of the 26th international conference on world wide web*, pp. 173–182 (cited on pp. 33, 34, 64, 81, 82).

Herlocker, Jon, Joseph A Konstan, and John Riedl (2002). "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms". In: *Information retrieval* 5, pp. 287–310 (cited on p. 31).

Herlocker, Jonathan L, Joseph A Konstan, Loren G Terveen, and John T Riedl (2004). "Evaluating collaborative filtering recommender systems". In: *ACM Transactions on Information Systems (TOIS)* 22.1, pp. 5–53 (cited on p. 40).

Hoffart, Johannes, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum (2011). "YAGO2: exploring and querying world knowledge in time, space, context, and many languages". In: *Proceedings of the 20th international conference companion on World wide web*, pp. 229–232 (cited on p. 20).

Hofmann, Thomas (1999). "Probabilistic Latent Semantic Analysis". In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. UAI'99. Stockholm, Sweden: Morgan Kaufmann Publishers Inc., 289–296. ISBN: 1558606149 (cited on p. 39).

Hogan, Aidan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. (2021). "Knowledge graphs". In: *ACM Computing Surveys (CSUR)* 54.4, pp. 1–37 (cited on pp. 19–25).

Horton, William (2011). *E-learning by design*. John Wiley & Sons (cited on pp. 3, 11).

Horvat, Ana, Marina Dobrota, Maja Krsmanovic, and Mladen Cudanov (2015). "Student perception of Moodle learning management system: a satisfaction and significance analysis". In: *Interactive Learning Environments* 23.4, pp. 515–527 (cited on p. 14).

Huang, Xiao, Jingyuan Zhang, Dingcheng Li, and Ping Li (2019). "Knowledge graph embedding based question answering". In: *Proceedings of the twelfth ACM international conference on web search and data mining*, pp. 105–113 (cited on p. 22).

Hussein, Rana, Dingqi Yang, and Philippe Cudré-Mauroux (2018). "Are meta-paths necessary? Revisiting heterogeneous graph embeddings". In: *Proceedings of the 27th ACM international conference on information and knowledge management*, pp. 437–446 (cited on p. 20).

Ilkou, Eleni (2022). "Personal knowledge graphs: use cases in e-learning platforms". In: *Companion Proceedings of the Web Conference 2022*, pp. 344–348 (cited on p. 19).

Jethro, Olojo Oludare, Adewumi Moradeke Grace, and Ajisola Kolawole Thomas (2012). "E-learning and its effects on teaching and learning in a global age". In: *International Journal of Academic Research in Business and Social Sciences* 2.1, p. 203 (cited on p. 11).

Ji, Guoliang, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao (2015). "Knowledge graph embedding via dynamic mapping matrix". In: *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pp. 687–696 (cited on pp. 23, 42, 63).

Ji, Shaoxiong, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip (2021). "A survey on knowledge graphs: Representation, acquisition, and applications". In: *IEEE transactions on neural networks and learning systems* 33.2, pp. 494–514 (cited on p. 5).

Jing, Xia and Jie Tang (2017). "Guess you like: course recommendation in MOOCs". In: *Proceedings of the international conference on web intelligence*, pp. 783–789 (cited on p. 45).

Kabbur, Santosh, Xia Ning, and George Karypis (2013). "Fism: factored item similarity models for top-n recommender systems". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 659–667 (cited on p. 82).

Kazemi, Seyed Mehran and David Poole (2018). "Simple embedding for link prediction in knowledge graphs". In: *Advances in neural information processing systems* 31 (cited on p. 25).

Kendal, Simon L and Malcolm Creen (2007). *An introduction to knowledge engineering*. Springer (cited on p. 16).

Kim, Tae dong, Min young Yang, Jinhwa Bae, Byoung a Min, Inseong Lee, and Jinwoo Kim (2017). "Escape from infinite freedom: Effects of constraining user freedom on the prevention of dropout in an online learning context". In: *Computers in Human Behavior* 66, pp. 217–231 (cited on p. 15).

Kim, Yoon (2014). "Convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1408.5882* (cited on p. 42).

Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (cited on p. 91).

Kuzilek, Jakub, Martin Hlosta, and Zdenek Zdrahal (2017). "Open university learning analytics dataset". In: *Scientific data* 4.1, pp. 1–8 (cited on pp. 69–72).

Lehmann, Jens, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. (2015). "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia". In: *Semantic web* 6.2, pp. 167–195 (cited on p. 20).

Li, Junling and Wanyu Che (2022). "Challenges and coping strategies of online learning for college students in the context of COVID-19: A survey of Chinese universities". In: *Sustainable Cities and Society* 83, p. 103958. ISSN: 2210-6707 (cited on p. 3).

Liang, Chen, Zhaohui Wu, Wenyi Huang, and C Lee Giles (2015). "Measuring prerequisite relations among concepts". In: *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1668–1674 (cited on p. 71).

Liang, Hong, Xiao Sun, Yunlei Sun, and Yuan Gao (2017). "Text feature extraction based on deep learning: a review". In: *EURASIP journal on wireless communications and networking* 2017.1, pp. 1–12 (cited on p. 59).

Lin, Huann-shyang, Zuway-R Hong, and Frances Lawrenz (2012). "Promoting and scaffolding argumentation through reflective asynchronous discussions". In: *Computers & Education* 59.2, pp. 378–384 (cited on p. 12).

Lin, Yankai, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu (2015). "Learning entity and relation embeddings for knowledge graph completion". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 29. 1 (cited on pp. 23, 43).

Lin, Yuanguo, Shibo Feng, Fan Lin, Wenhua Zeng, Yong Liu, and Pengcheng Wu (2021). "Adaptive course recommendation in MOOCs". In: *Knowledge-Based Systems* 224, p. 107085 (cited on p. 45).

Liu, Qi, Matt J Kusner, and Phil Blunsom (2020). "A survey on contextual embeddings". In: *arXiv preprint arXiv:2003.07278* (cited on p. 59).

Liu, Qiang, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang (2016). "Context-aware sequential recommendation". In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, pp. 1053–1058 (cited on p. 40).

Liu, Tieyuan, Qiong Wu, Liang Chang, and Tianlong Gu (2022). "A review of deep learning-based recommender system in e-learning environments". In: *Artificial Intelligence Review* 55.8, pp. 5953–5980 (cited on pp. 45, 46).

Liu, Zheng, Xing Xie, and Lei Chen (2018). "Context-aware academic collaborator recommendation". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1870–1879 (cited on p. 44).

Lops, Pasquale, Marco De Gemmis, and Giovanni Semeraro (2011). "Content-based recommender systems: State of the art and trends". In: *Recommender systems handbook*, pp. 73–105 (cited on p. 37).

Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey (2015). "Adversarial autoencoders". In: *arXiv preprint arXiv:1511.05644* (cited on p. 60).

Martin, Florence, Ting Sun, and Carl D Westine (2020). "A systematic review of research on online teaching and learning from 2009 to 2018". In: *Computers & education* 159, p. 104009 (cited on pp. 3, 11).

Matviichuk, Liudmyla, Stefano Ferilli, and Nataliia Hnedko (2022). "Study of the Organization and Implementation of E-Learning in Wartime Inside Ukraine". In: *Future Internet* 14.10, p. 295 (cited on p. 3).

Melville, Prem, Raymond J Mooney, Ramadass Nagarajan, et al. (2002). "Content-boosted collaborative filtering for improved recommendations". In: *Aaai/iaai* 23, pp. 187–192 (cited on p. 39).

Melville, Prem and Vikas Sindhwani (2010). "Recommender systems." In: *Encyclopedia of machine learning* 1, pp. 829–838 (cited on pp. 34, 89).

Miller, Justin J (2013). "Graph database applications and concepts with Neo4j". In: *Proceedings of the southern association for information systems conference, Atlanta, GA, USA*. Vol. 2324. 36 (cited on pp. 21, 22).

Navigli, Roberto and Simone Paolo Ponzetto (2012). "BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network". In: *Artificial intelligence* 193, pp. 217–250 (cited on p. 20).

Negre, Elsa (2015). *Information and recommender systems*. John Wiley & Sons (cited on p. 27).

Nickel, Maximilian, Lorenzo Rosasco, and Tomaso Poggio (2016). "Holographic embeddings of knowledge graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1 (cited on p. 24).

Nickel, Maximilian and Volker Tresp (2013). "Tensor factorization for multi-relational learning". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*. Springer, pp. 617–621 (cited on p. 24).

Norén Creutz, Isabella and Matilda Wiklund (2014). "Learning paradigms in workplace e-learning research". In: *Knowledge Management & E-Learning: An International Journal* 6.3, pp. 299–315 (cited on p. 11).

Noy, Natalya F, Deborah L McGuinness, et al. (2001). *Ontology development 101: A guide to creating your first ontology* (cited on pp. 6, 51, 90).

Noy, Natasha, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor (2019). "Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done". In: *Queue* 17.2, pp. 48–75 (cited on p. 19).

Paulheim, Heiko (2017). "Knowledge graph refinement: A survey of approaches and evaluation methods". In: *Semantic web* 8.3, pp. 489–508 (cited on pp. 20, 22).

Pazzani, Michael J (1999). "A framework for collaborative, content-based and demographic filtering". In: *Artificial intelligence review* 13, pp. 393–408 (cited on p. 39).

Piotrowski, Michael (2010). "What is an e-learning platform?" In: *Learning management system technologies and software solutions for online teaching: Tools and applications*. IGI Global, pp. 20–36 (cited on pp. 3, 13, 14).

Popescul, Alexandrin, Lyle H Ungar, David M Pennock, and Steve Lawrence (2013). "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments". In: *arXiv preprint arXiv:1301.2303* (cited on p. 39).

Porter, Gavin W (2013). "Free choice of learning management systems: Do student habits override inherent system quality?" In: *Interactive Technology and Smart Education* 10.2, pp. 84–94 (cited on p. 14).

Pujara, Jay, Hui Miao, Lise Getoor, and William Cohen (2013). "Knowledge graph identification". In: *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I 12*. Springer, pp. 542–557 (cited on p. 20).

Rajeh, Mona T, Fahad H Abduljabbar, Saad M Alqahtani, Feras J Waly, Ibrahim Alnaami, Abdulaziz Aljurayyan, and Naweed Alzaman (2021). "Students' satisfaction and continued intention toward e-learning: A theory-based study". In: *Medical education online* 26.1, p. 1961348 (cited on p. 12).

Rendle, Steffen, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme (2014). "Bayesian personalized ranking from implicit feedback".

In: *Proc. of Uncertainty in Artificial Intelligence*, pp. 452–461 (cited on p. 81).

Resnick, Paul, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl (1994). "Grouplens: An open architecture for collaborative filtering of netnews". In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp. 175–186 (cited on p. 30).

Said, Ghada Refaat El (2017). "Understanding How Learners Use Massive Open Online Courses and Why They Drop Out: Thematic Analysis of an Interview Study in a Developing Country". In: *Journal of Educational Computing Research* 55.5, pp. 724–752 (cited on p. 16).

Salton, Gerard (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201122278 (cited on p. 35).

Sanchez-Lengeling, Benjamin, Emily Reif, Adam Pearce, and Alexander B Wiltschko (2021). "A gentle introduction to graph neural networks". In: *Distill* 6.9, e33 (cited on p. 91).

Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl (2000). *Application of dimensionality reduction in recommender system-a case study*. Tech. rep. Minnesota Univ Minneapolis Dept of Computer Science (cited on p. 32).

Scarselli, Franco, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini (2008). "The graph neural network model". In: *IEEE transactions on neural networks* 20.1, pp. 61–80 (cited on p. 91).

Shardanand, Upendra and Pattie Maes (1995). "Social information filtering: Algorithms for automating "word of mouth"". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 210–217 (cited on p. 38).

Sheth, Beerud and Pattie Maes (1993). "Evolving agents for personalized information filtering". In: *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*. IEEE, pp. 345–352 (cited on p. 39).

Shi, Daqian, Ting Wang, Hao Xing, and Hao Xu (2020). "A learning path recommendation model based on a multidimensional knowledge graph framework for e-learning". In: *Knowledge-Based Systems* 195, p. 105618 (cited on p. 44).

Shu, Jiangbo, Xiaoxuan Shen, Hai Liu, Baolin Yi, and Zhaoli Zhang (2018). "A content-based recommendation algorithm for learning resources". In: *Multimedia Systems* 24.2, pp. 163–173 (cited on p. 45).

Sielis, George A, Aimilia Tzanavari, and George A Papadopoulos (2015). "Recommender systems review of types, techniques, and applications". In: *Encyclopedia of Information Science and Technology, Third Edition*. IGI global, pp. 7260–7270 (cited on p. 49).

Socher, Richard, Danqi Chen, Christopher D Manning, and Andrew Ng (2013). "Reasoning with neural tensor networks for knowledge base

completion". In: *Advances in neural information processing systems* 26 (cited on p. 25).

Srivastava, Nitish and Russ R Salakhutdinov (2012). "Multimodal learning with deep boltzmann machines". In: *Advances in neural information processing systems* 25 (cited on p. 64).

Steck, Harald (2013). "Evaluation of recommendations: rating-prediction and ranking". In: *Proceedings of the 7th ACM conference on Recommender systems*, pp. 213–220 (cited on p. 40).

Stecuła, Kinga and Radosław Wolniak (2022a). "Advantages and Disadvantages of E-Learning Innovations during COVID-19 Pandemic in Higher Education in Poland". In: *Journal of Open Innovation: Technology, Market, and Complexity* 8.3, p. 159 (cited on p. 12).

— (2022b). "Influence of COVID-19 pandemic on dissemination of innovative e-learning tools in higher education in Poland". In: *Journal of Open Innovation: Technology, Market, and Complexity* 8.2, p. 89 (cited on p. 12).

Sun, Zhiqing, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang (2019). "Rotate: Knowledge graph embedding by relational rotation in complex space". In: *arXiv preprint arXiv:1902.10197* (cited on p. 23).

Tang, Qing, Marie-Hélène Abel, and Elsa Negre (2021). "Incorporating Dynamic Information into Content-Based Recommender System in Online Learning Environment". In: *International Conference on Deep Learning, Artificial Intelligence and Robotics*. Springer, pp. 97–106 (cited on p. 37).

— (2022). "Personalized Services in Collaborative Learning Environment Based on Learner's Activity Records". In: *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, pp. 1420–1425 (cited on pp. 18, 53, 54).

Tayebinik, Maryam and Marlia Puteh (2013). "Blended Learning or E-learning?" In: *arXiv preprint arXiv:1306.4085* (cited on pp. 3, 11).

Trouillon, Théo, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard (2016). "Complex embeddings for simple link prediction". In: *International conference on machine learning*. PMLR, pp. 2071–2080 (cited on p. 24).

Vashishth, Shikhar, Prince Jain, and Partha Talukdar (2018). "Cesi: Canonicalizing open knowledge bases using embeddings and side information". In: *Proceedings of the 2018 World Wide Web Conference*, pp. 1317–1327 (cited on p. 22).

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio (2017). "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (cited on p. 91).

Vrandečić, Denny and Markus Krötzsch (2014). "Wikidata: a free collaborative knowledgebase". In: *Communications of the ACM* 57.10, pp. 78–85 (cited on p. 20).

Wang, Hongwei, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo (2018a). "Ripplenet: Propagating user preferences

on the knowledge graph for recommender systems". In: *Proceedings of the 27th ACM international conference on information and knowledge management*, pp. 417–426 (cited on p. 42).

Wang, Hongwei, Fuzheng Zhang, Xing Xie, and Minyi Guo (2018b). "DKN: Deep knowledge-aware network for news recommendation". In: *Proceedings of the 2018 world wide web conference*, pp. 1835–1844 (cited on p. 42).

Wang, Hongwei, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo (2019a). "Multi-task feature learning for knowledge graph enhanced recommendation". In: *The world wide web conference*, pp. 2000–2010 (cited on p. 43).

Wang, Minhong, Weijia Ran, Jian Liao, and Stephen JH Yang (2010). "A performance-oriented approach to e-learning in the workplace". In: *Journal of Educational Technology & Society* 13.4, pp. 167–179 (cited on p. 11).

Wang, Quan, Zhendong Mao, Bin Wang, and Li Guo (2017). "Knowledge graph embedding: A survey of approaches and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 29.12, pp. 2724–2743 (cited on pp. 20, 22, 25, 62, 63, 91).

Wang, Xiao, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu (2019b). "Heterogeneous graph attention network". In: *The world wide web conference*, pp. 2022–2032 (cited on p. 20).

Wang, Zhen, Jianwen Zhang, Jianlin Feng, and Zheng Chen (2014). "Knowledge graph embedding by translating on hyperplanes". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 28. 1 (cited on pp. 20, 23).

Willmott, Cort J and Kenji Matsuura (2005). "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance". In: *Climate research* 30.1, pp. 79–82 (cited on p. 40).

Wu, Lingfei, Peng Cui, Jian Pei, Liang Zhao, and Xiaojie Guo (2022). "Graph neural networks: foundation, frontiers and applications". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4840–4841 (cited on p. 91).

Yang, Bishan, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng (2014). "Embedding entities and relations for learning and inference in knowledge bases". In: *arXiv preprint arXiv:1412.6575* (cited on p. 24).

Yang, Luwei, Zhibo Xiao, Wen Jiang, Yi Wei, Yi Hu, and Hao Wang (2020). "Dynamic heterogeneous graph embedding using hierarchical attentions". In: *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42*. Springer, pp. 425–432 (cited on p. 20).

Yu, Jifan, Gan Luo, Tong Xiao, Qingyang Zhong, Yuquan Wang, Wenzheng Feng, Junyi Luo, Chenyu Wang, Lei Hou, Juanzi Li, et al. (2020).

"MOOCCube: a large-scale data repository for NLP applications in MOOCs". In: *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 3135–3142 (cited on pp. 69, 70).

Yu, Jifan, Chenyu Wang, Gan Luo, Lei Hou, Juanzi Li, Jie Tang, and Zhiyuan Liu (2019). "Course concept expansion in moocs with external knowledge and interactive game". In: *arXiv preprint arXiv:1909.07739* (cited on p. 71).

Zhang, Fuzheng, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma (2016). "Collaborative knowledge base embedding for recommender systems". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 353–362 (cited on pp. 22, 43).

Zhang, Hanwang, Yang Yang, Huanbo Luan, Shuicheng Yang, and Tat-Seng Chua (2014). "Start from scratch: Towards automatically identifying, modeling, and naming visual attributes". In: *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 187–196 (cited on p. 64).

Zhang, Huanyu, Xiaoxuan Shen, Baolin Yi, Wei Wang, and Yong Feng (2023). "KGAN: Knowledge grouping aggregation network for course recommendation in MOOCs". In: *Expert Systems with Applications* 211, p. 118344 (cited on p. 44).

Zhang, Jing, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun (2019a). "Hierarchical reinforcement learning for course recommendation in MOOCs". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01, pp. 435–442 (cited on pp. 45, 71).

Zhang, Shuai, Lina Yao, Aixin Sun, and Yi Tay (2019b). "Deep learning based recommender system: A survey and new perspectives". In: *ACM computing surveys (CSUR)* 52.1, pp. 1–38 (cited on pp. 4, 27, 29, 32, 89).

Zhang, Yin, Rong Jin, and Zhi-Hua Zhou (2010). "Understanding bag-of-words model: a statistical framework". In: *International journal of machine learning and cybernetics* 1, pp. 43–52 (cited on p. 60).

Zhou, Dongzhuoran, Baifan Zhou, Zhuoxun Zheng, Ahmet Soylu, Gong Cheng, Ernesto Jimenez-Ruiz, Egor V Kostylev, and Evgeny Kharlamov (2022). "Ontology reshaping for knowledge graph construction: Applied on bosch welding case". In: *The Semantic Web–ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings*. Springer, pp. 770–790 (cited on p. 20).

Zhou, Meizi, Zhuoye Ding, Jiliang Tang, and Dawei Yin (2018). "Micro behaviors: A new perspective in e-commerce recommender systems". In: *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 727–735 (cited on p. 40).