Research on High School Math Exercise Recommendation Based on Graph Neural Network

Anonymous

Supervisor: Anonymous

This thesis is submitted for the degree of
Master of Computer Science

Central China Normal University Wollongong Joint Institute
Central China Normal University
April 2021

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This thesis is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments.

Anonymous
April 2021

Abstract

There are difficulties in evaluating students' learning status in high school Math education and recommending appropriate learning resources. High school mathematics exercise recommendation system as an adaptive learning system is intended to solve this problem by tracking students' knowledge mastery proficiency and recommending appropriate practicing exercises.

Based on this research background, a knowledge-tracing-based exercise recommendation system is designed and proposed. It consists of three modules: the exercise knowledge point labeling module as the pre-processing part, the knowledge tracing module as the core, and the exercise recommendation module as the functional part.

1. In the exercise knowledge point labeling model, a new network combining graph neural network and attention-based bidirectional LSTM layer is proposed for knowledge point labeling of exercise, which achieves better performance in classifying hidden knowledge points and knowledge point with complicated relations in experiments comparing with several baseline models.

2. In the knowledge tracing module, an improved model for the DKVMN knowledge tracing network is proposed, adapting the node information propagation mechanism of graph neural networks into the network's memory structure and integrating students' answering behaviors as an extra feature. The experiments showed that the proposed model outperforms original DKVMN models and other baseline models in specific metrics.

3. In the exercise recommendation module, a recommendation model based on matching and ranking is proposed. The model applies different matching strategies to generate candidate recommendation sets and then uses the knowledge mastery proficiency output from the knowledge tracking model to rank recommended exercises. Several experiments are performed to validate the effectiveness of the model.

After theoretical derivation and experimental verification, the proposed model meets the expected objectives and design requirements.

Keywords: Graph neural network, Knowledge point labeling, Knowledge tracing, Exercise recommendation

摘 要

2010 年后，人工智能技术逐渐成为计算机技术领域的研究热点。尤其是机器围棋手 AlphaGo 的问世，引发了业界对于人工智能前景的极大关注。这带来行业的爆发式发展，大量的研究课题被提出。在人工智能相关研究中，各种算法创新、理论突破和模型应用层出不穷，为各个行业的智能化奠定了基础。人工智能技术在教育领域的应用也催生了智能教育概念的出现。其中，自适应学习是智能教育中的热门的应用领域之一。自适应学习模型一般是通过结合对海量学生群体学习数据的大数据分析和对目标学生个体数据的精准化数据分析来追踪学生的学习状态，从而针对学生的个体特征和知识掌握熟练度来生成个性化学习路径。自适应学习技术可以将以往需要大量人工劳动的学生评估和教学计划等工作，通过自动化机器学习算法来完成，这可以系统性缓解目前国内教育资源稀缺和分配不均的问题，也可以减轻教育从业者和学生的负担。因而它具有极大的的发展前景和商业价值，市面上也有越来越多的人工智能研究团队和智能化教育技术公司专注于自适应学习系统的研发和应用。部分智能教育科技公司已开始将自适应学习用作其产品要核心功能或主要卖点。自适应教育技术可以综合分析学生个体层面的学习能力、知识熟练度和群体层面的热门学习资源、易错题等，从而可以将最适合的学习路径和学习资源例如习题、资料、知识点推送给学生。系统会根据学生的知识状态自动调整推送学习资源的知识侧重点，防止重复练习已经掌握的知识点或者缺乏练习未掌握的知识点。一方面，教师可以根据系统输出的数据或可视化图表来制作每个学生的学习状态评估报告分析整个班级的所有学生的知识掌握熟练度，适应性地调整总体教学计划。另一方面，学生可以通过系统来分析自己的知识薄弱项，从而针对性的进行习题训练。因此，适应性学习是在线教育中“学生知识掌握状态自动评估和教学方案生成”问题潜在解决方案之一。

本文以高中数学学科为主要研究背景，目的是提出一种基于知识追踪的高中数学习题推荐模型。在高中数学学科中，练习习题为学生主要的学习能力提高手段。但是目前高中数学教学中，教师或学生需要从庞大的习题库中去寻找合适的习题进行练习，它们往往存在过于庞大、重复度高和组织混乱等问题。在习题库中存在相当多低质量的未标注知识点的习题，需要人工进行知识点标注。有部分学生通过题海战术来进行学习，但这样效率较低，且往往出现熟悉知识点的重复练习和不熟悉知

识点的缺乏练习等情况。为了提高学生进行习题练习的效果，从而提升知识掌握的熟练度和全面性，需要经验丰富的教学人员进行学生知识状态分析，从习题库中筛选出合适的习题进行推荐。该方法人工工作量大，依赖专家先验知识，且包含大量的重复性工作，因此存在不经济且低效的问题。此外，传统习题推荐以学生群体为单位，没有针对特定学生的知识掌握情况进行推荐，也没有考虑不同学生的学习能力不同的问题，因此导致推荐的效果精细度较差。为了改善传统习题推荐方法存在的问题，可以通过应用知识追踪技术来追踪学生的学习情况，从而针对性地进行自动化习题推荐。本文的目标在于设计一个基于知识点标注、知识追踪和资源推荐技术的习题推荐系统，从而推出一个在习题推荐方面的智能自适应学习的解决方案。

本文提出的高中数学习题推荐系统包括三个模块，分别为习题知识点标注模块、知识追踪模块和推荐模块。习题知识点标签模块是习题推荐的前置工作，其作用是为未标注知识点的习题进行知识点标注，从而将传统的人工知识点标注以自动化的形式代替。经过知识标注的习题可以作为习题推荐系统的数据源。知识追踪是整个系统的核心部分，通过追踪学生的习题练习记录，计算学生的知识熟练度状态向量，它是学生对于学科知识点、概念和技能的掌握熟练度的表征。习题推荐模块是系统的功能模块，具有召回和排序两个阶段，前一阶段于原始习题库上应用多种召回策略对习题进行快速筛选，生成推荐候选习题集合，后一阶段输入该集合在排序阶段输入知识追踪系统中进行精细化推荐排序，生成最终的推荐结果。

- 第二章提出了一种结合注意力机制的双向 LSTM 文本信息表征网络与图神经网络知识点关系表征网络的高中数学习题多知识点标注方法。习题知识点标注模块包含习题文本挖掘和多知识点标签分类两个子模块。由于习题库的大多数习题只包含文本信息等非结构化数据，因此本文主要通过习题文本挖掘的方式来进行知识点提取。它应用了加入注意力机制的双向 LSTM 网络来进行习题文本挖掘，习题首先经过文本清洗、去重等预处理步骤，在得到词序列的同时过滤掉大量的无关信息的干扰。然后利用 BERT 来作为词向量的嵌入表示，可以表征词向量间的隐藏依赖关系，这有利于构建知识点间依赖关系。之后，通过双向 LSTM 模型进行文本信息抽取，能够更好地解决文本中上下文元素长程依赖的问题，生成习题文本表征向量。另外，为了在分类模型上捕捉知识点间依赖关系，本文提出了一个基于图卷积神经网络（GCN）的多标签知识点标注模型，每个标签都作为图上的一个节点表示，经过多轮迭代学习，将标签图映射为一组内在依赖的知识点分类器。随后，将提取的习题文本表征向量输入知识点分类器组，得出多知识点预测概率向量，从而实现多知识点标签标注任务。实验阶段，通过在自制的高中数学习题数据集上进行实验，将本论文提出的方法与一系列基准模型进行对比，并采用一系列多标签分类指标来进行模型性能比较和评估。实验结果显示该方法相对于其他的基线模型取得了更好的性能。

- 第三章提出了一种针对动态键值记忆网络（DKVMN）的改进模型。该模型继承了原始 DKVMN 模型的基于知识点权重计算习题和学生掌握相关度的思想，相对于原始的 DKVMN 模型有如下改进。模型的第一个改进是尝试在模型中融入答题延迟、请求提示等学生答题特征，从而捕捉学生的个性化特征对于习题解答的影响。模型的第二个改进点是应用图神经网络结构到键值存储模块上，以引入相关存储单元的相互影响。这增强了原始模型对于相关知识点的表征能力，在对模型进行知识点掌握熟练度进行修正的过程中，运用图网络相邻节点传播机制，对相关知识点重新调整熟练度变化。在实验阶段，在公开的数据集上与原始 DKVMN 模型和一些其他的基线模型进行多方面的对比。实验表明，模型的性能和可解释性相对于原始的 DKVMN 模型以及其他的基线模型有一定的提升。

- 第四章提出了基于召回-排序两阶段的数学习题推荐模型。第一阶段为召回模型，它是一个基于多召回策略的混合模型，它具有多路召回和融合两个过程。在多路召回过程，采用了基于协同过滤、热门度、用户偏好等多个召回策略用于分别生成习题推荐候选集合。然后在融合过程，将这些候选集合进行加权排序合并，形成一个最终的习题推荐候选集合。第二个阶段为基于知识追踪的推荐项排序模型，将前一阶段获取的习题候选集合中的习题输入到前一章提出的知识追踪模型，进行正确率预测，将最容易出错的习题作为优先级最高的推荐项。该模型结合了知识追踪模型的输出，能够将学生容易出错的习题作为最终推荐结果，能够起到查漏补缺效果。经过在公开数据集上的性能测试和与基线模型的对照实验，结果显示提出的模型可以有效预测合适的习题来完成推荐。

综上，通过分析系统的需求，将整个推荐系统合理化地分为多个模块，针对各个模块设计了不同的模型和算法来实现系统的功能，并设计了相关的实验来验证模型的有效性和先进性。经过实验验证，提出的模型能够达到各项预期设计指标。

关键词： 图神经网络，知识点标注，知识追踪，习题推荐

Acknowledgments

After two years of study at the master's level, I sincerely express my sincere gratitude to my teachers, faculty leaders, counselors, classmates, and friends who have helped me on the occasion of completing my dissertation.

First of all, I would like to give special thanks to my supervisor, Prof. Wang, who has given me patient and meticulous guidance and a great help in the process of completing my usual scientific tasks. He also provided many valuable and enlightening suggestions while selecting the topic, conception, experimental simulation, and thesis writing of this paper. Finally, my thesis has been improved continuously. This is all thanks to Mr. Wang's diligent guidance and teachings.

Then, I would like to thank all the college leaders and teachers for their considerable and valuable advice during my work and study at the master's level, which has refined my heart and for which I am full of gratitude. This is undoubtedly an invaluable asset for my future career.

I would also like to thank all my classmates and friends who have walked with me through my research study career in the past three years, and we have helped each other in our study, motivated and cared for each other. Thanks for their friendship and support!

This work was supported by the National Natural Science Foundation of China (No.61901165, 61501199), Science and Technology Research Project of Hubei Education Department (No. Q20191406), Hubei Natural Science Foundation (No. 2017CFB683), Hubei Research Center for Educational Informationization Open Funding (No. HRCEI2020F0102), and Self-determined Research Funds of CCNU from the Colleges' Basic Research and Operation of MOE (No. CCNU20ZT010).

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Background and Significance

The development of technological research and commercial application deployment of artificial intelligence has shown an accelerating trend in recent years. The deployment of various algorithms related to AI and big data analytics based on artificial intelligence plays a positive role in accelerating enterprises and organizations' digitization, improving industry chains' structure, and increasing information utilization efficiency. As a traditional service industry, the education industry also has considerable prospect for an intelligent revolution. In traditional education models, student groups are often dividded into basic education units like classes. Therefore, in a class, the granularity of all teaching activities is equivalent to the class size, which leads to the fact that the content of students' learning does not entirely match their needs. The knowledge points that the students have mastered are over-practice, or the knowledge points that the students have not mastered lack practice. This has caused students to be tired of learning, anxiety, and other conditions, significantly impacting their learning efficiency and learning effect. On the other hand, for teachers, generalized detailed monitoring and evaluation of individual knowledge status require a considerable workload, so teachers often only pay attention to a part of the students, leading to the neglect of most students' learning situation, which has a more significant impact on students' learning enthusiasm and learning conditions.

In recent years, China has released a series of policies to promote artificial intelligence in education. AI technology for student learning monitoring and tracking can free up teachers' labor and solve problems that are difficult to solve in traditional manual education, thus significantly improving the quality and efficiency of education and realizing smart education. In smart education, adaptive learning is a model that has been proven in practice, and it already has many successful cases of commercial deployment [1]. A considerable number of online education platforms have deployed adaptive learning education system services in different domains. It uses data analytics, machine learning, and automation to automatically assess and track students' knowledge status by analyzing their learning behavior data. Besides, the system provides students with learning path planning and learning resource recommendation services by combining personalized information such as students' potential and areas of expertise [2]. It can improve teachers' teaching quality while reducing teachers' work pressure and effectively improve their learning efficiency. The exercise recommendation system implements an adaptive learning model, including learner knowledge mastery proficiency modeling and exercise recommendation. The general model of knowledge mastery modeling is to input the semantic records of learners' learning interactions such as question records, quiz records to capture learners' learning characteristics and achieve the dynamic tracking of their knowledge mastery proficiency. The exercise recommendation part analyzes the learners' knowledge mastery proficiency model and recommends exercises relevant to the learners' relatively weak knowledge mastery.

High school mathematics is a relatively tricky subject at the high school level. In high school mathematics, the knowledge points are complicated and closely interconnected, and the corresponding library of exercises is extensive and confusing. As a result, many students do not know how to reasonably assess their knowledge and practice in a targeted manner, leading to the emergence of "excessive practicing tactics", which hurts students' interest in learning and self-confidence. The purpose of this thesis is to propose a system for tracking students' knowledge of mathematics in real-time and recommending exercises according to their knowledge status. The system is divided into three parts, the first part is the knowledge labeling of the exercises as the data preparation part of the exercise recommendation system, the second part is the core knowledge tracing model, which tracks the knowledge proficiency status by inputting students' records of doing exercises, and the third part is the functional part of the system, which recommends exercises targeted by students' knowledge proficiency output from the knowledge tracing model. The system can effectively achieve the goal of adaptive learning.

1.2 Research Status

This thesis's research topic is a recommendation system for high school mathematics exercises, for which there have been several studies and applications. A popular approach is to evaluate and track students' knowledge mastery status using cognitive diagnosis and then recommend corresponding exercises based on students' knowledge status. For example, wang et al. proposed NeuralCD [3], a cognitive diagnostic model combined with neural networks, which has improved in both prediction accuracy and interpretability. There are also some improvements to the model, such as Huang et al. [4] applying multilevel structure into higher-order latent features to extend the original model into a multilevel higher-order cognitive diagnostic model.

The first part of this thesis is a graph neural network and Attentional Bi-LSTM based knowledge point labeling for high school math exercises, the second part is a graph attention network and Transformer based knowledge tracing model, and the third part is a matching and ranking based exercise recommendation module. Thus, the techniques covered in this thesis include high school mathematics subjects, graph neural networks, natural language processing, multi-label classification, recommendation systems, and other research topics. Next, this section reviews the current state of research on these techniques.

1.2.1 High School Mathematics

Mathematics is an essential subject in both primary education and scientific research. Mathematics is a science devoted to studying relationships between quantities and spatial forms, with a complete system of symbols, a clear and unique structure of formulas, and more vivid and intuitive verbal expressions such as words and images. In mathematics, the establishment of knowledge structure and cognitive structure plays a considerable role in learning the subject. The "cognitive structure" denotes the organization of declarative knowledge among the human brain, while the cognitive structure internalized through learning and displayed through network structures or graphics is the knowledge structure [5]. Most of the knowledge contents that learners need to learn come from the experience summaries of previous people in practical activities, and the process of their learning is the cognitive learning of this summarized knowledge, and constantly digesting, adjusting, and reorganizing the structure of knowledge, to build a more perfect and suitable knowledge structure, which is also a process of combining with innovative thinking. In mathematics, knowledge mastery often comes from practice, such as exercises, proofs, and derivations. As a primary subject at the secondary level, mathematics is characterized by a high degree of abstraction, rigorous logic, and extensive applications. The body of knowledge in this subject is built up using many abstract concepts, and new abstract concepts are formed by learning and expanding on these concepts. Also, mathematics is logical because any conclusion reached in mathematics requires rigorous logical reasoning and strict proof before it can be considered reasonable. It is an essential tool for social practice or scientific research, and the study of mathematics is indispensable in all walks of life and all areas of society. Mathematical knowledge is also intrinsically related to each other to be arranged and learned in a specific logical progression in the specific learning process. These intrinsic relationships can be classified as synonymous, antecedent, successor, inclusion, brotherhood, opposition, and so on. A network of knowledge point associations can be established to facilitate subsequent knowledge mastery status tracking by analyzing the knowledge points.

1.2.2 Graph Neural Network

In recent years, the rapid development of neural network models has driven research related to machine learning, and various neural network paradigms have been designed for various application environments and tasks, such as convolutional neural networks (CNNs), which are widely used in the pattern and feature recognition, and recurrent neural networks (RNNs), which are applied to serialized data learning. Traditional neural networks have good computational and processing capabilities for data in Euclidean space such as language, sequences, and images, but have limitations for non-Euclidean space such as knowledge networks. In this thesis, the study object is mathematical subject knowledge, and the knowledge points form a net-like correlation between them. Therefore, the complex relationship between knowledge points cannot be fully characterized by traditional neural network model processing, and it is more compatible with the natural paradigm to learn the knowledge point network by the graph. Each knowledge point or exercise can be treated as a node of a graph, and the edges between nodes represent the association between knowledge points or exercises. Therefore, this thesis introduces a graph neural network to capture the correlation between knowledge points and exercises, which can reasonably abstract the data in a higher dimension and achieve better results.

At present, for the graph structure, the machine learning tasks that can be performed can be roughly divided into the following types:

1. Graph node classification task: For a graph in which each node has a corresponding feature, and the categories of some nodes are known, a classification task can be designed to classify unknown nodes.

2. Graph edge structure prediction task: For a graph where the edge relationship between some nodes is known, the edge structure and relationship of the location are mined based on the existing information. This type of task is the edge prediction task: Prediction of the relationship between nodes and nodes.

3. Graph classification: graph classification is also called the graph isomorphism problem. This is often achieved by aggregating the node characteristics of the graph and then classifying it.

The graph neural network model is based on the theory of immobile points and aims to obtain the hidden state of each node [6]. However, when there are too many stacks, the state convergence tends to be too smooth and challenging to learn the graph's feature information. Graph neural networks are generally applied iteratively to compute data transfer to nodes as one method to learn the target node representation. The concept and application of graph neural networks have undergone continuous development, and new graph neural network models have been proposed one after another. In recent years, the increase in computing power of parallel computing devices such as GPUs has made it possible to apply many graph neural network models that were too limited by computing power. In this thesis, a network structure similar to graph convolutional neural networks (GCN) is applied. GCN is a model proposed by Kipf in 2016 for semi-supervised classification [7], and the computation of GCN is based on a hierarchical structure, where each layer is the result of feature extraction from the previous layer, from the node level, and the nodes propagate each other hiding the state. The final GCN output undergoes multiple layers of abstraction, and in terms of learning of node feature information and graph structure information, GCN achieves State of the Art (SOTA) performance on almost all of the datasets related to most public node classification or edge prediction. GAT was proposed by Veličković et al. in 2018 [8]. The network introduces a self-attentive mechanism in the propagation process, where the hidden state of each node is computed by paying attention to its neighboring nodes. The network uses a local network design structure so that only neighboring nodes are computed during the computation, reducing the computational load.

1.2.3 Knowledge Tracing Algorithms

Knowledge tracing is a technology that models student knowledge acquisition based on past answer records and results. It is a classical model for modeling learner knowledge acquisition and has evolved into the mainstream approach for modeling learner knowledge acquisition in intelligent tutoring systems. The main task of knowledge tracing is to analyze learners' knowledge mastery based on their historical learning records and thus automatically track changes in students' knowledge levels over time in order to be able to accurately predict students' performance in future learning and provide appropriate learning tutoring. In this process, the knowledge space is used to describe the level of student knowledge acquisition. In the process of knowledge tracing, the knowledge space is modeled as a collection of concepts, and students' mastery of a portion of the collection of concepts constitutes students' mastery of knowledge. Some educational researchers argue that students' mastery of a specific set of related knowledge points affects their performance on exercises, i.e., the set of knowledge students have mastered closely related to their external performance on exercises. Teachers can assess students' knowledge status to understand better where their mastery is weak and target their instructional programs.s

In the knowledge tracing model, the traditional approach is implemented by cognitive diagnosis, which is an algorithm for modeling students' proficiency in knowledge points. Among them, Item response theory (IRT) [9] models learners based on a one-dimensional continuous model, while the DINA model models students' knowledge states based on a series of vectors that represent users' knowledge mastery related to the exercises [10]. Besides, Bayesian knowledge tracing (BKT) is a more widely used knowledge tracing model based on probabilistic graphical modeling [11], which models the learner's knowledge state as a set of binary variables representing whether the knowledge points are mastered or not, and the algorithm uses Hidden Markov Model (HMM) to maintain the knowledge proficiency variables. However, the drawback of BKT is that it ignores the forgetting property of students for knowledge and the correlation between knowledge points. In 2015, Deep Knowledge Tracing (DKT) was proposed [12], which is the first algorithm to apply RNN models to the knowledge tracing task, and the model uses LSTM to track the dynamic change of students' knowledge proficiency over time and learn the student's knowledge mastery vector, which is used to predict students' performance on questions.

Moreover, the Dynamic Key-Value Memory Networks (DKVMN) for knowledge tracing model proposed in 2017 borrows the idea of memory-enhanced networks, which can store students' mastery levels of knowledge concepts using a fundamental matrix by using the relationships between the underlying concepts, so the model can Explicitly output the student's mastery of the knowledge point. The model defines the relationship between exercises and knowledge points and achieves better performance than DKT and BKT on public datasets. However, these models do not appropriately model the complex correlation of knowledge points and degrade the prediction performance for exercises with complex knowledge point associations. In contrast, as a model adapted to model non-Euclidean spaces, a graph neural network can be a solution to model the relational network of knowledge points. The Graph-based knowledge tracing model proposed at ICLR2019 [13], which uses graph nodes to model student's responses to the exercises, considers the effect of doing questions on the knowledge state as well as similar exercises, and the test results in the public dataset tabulate the performance of this model over DKVMN.

1.2.4 Recommendation System

In the early days of the Internet, users often searched for content of interest. However, with the massive growth of Internet information, it has become difficult to search for suitable content. Therefore, it is based on pushing users the information they are interested in.

Recommendation system technology was developed. At present, the recommendation system has been widely used on the Internet, and it has brought considerable benefits to both service providers and users. In the field of education, for most students, the existing exercise database is too large, so students often experience information tragedy in the process of extracurricular exercises. The learning efficiency is relatively low. Therefore, a system for adaptive exercise recommendation based on students' knowledge status is necessary and important [14].

There are several types of algorithms in recommendation systems: collaborative filtering, content-based recommendation models, and so on. The core idea of a collaborative filtering algorithm is to divide users into groups based on similar interests, and according to this principle, to find users with similar interests to the recommended person and recommend the content that the group is interested in the user. The collaborative filtering algorithm integrates similar users' evaluation of recommended items to form a prediction model of users' interest in the item. Collaborative filtering is one of the most widely used and successful recommendation algorithms. Collaborative filtering can be further subdivided into two types: user-based model and item-based model [15]. The former focuses on finding similar users, i.e., recommendation subjects, and then recommending them through the content of interest to similar recommendation subjects. In contrast, the latter focuses on finding similar items, i.e., recommendation items, and filtering out the most similar recommendation items as recommendation results by calculating the item similarity. In collaborative filtering, the calculation of similarity is a key issue. There are already similarity algorithms such as Jaccard coefficient [16], cognitive similarity [17], Kullback-Leibler divergence based similarity [18], etc. Collaborative filtering often has a "cold boot" problem [19], i.e., when there are fewer similar users or entries, a large performance degradation occurs. Therefore, random recommendations or other content recommendation methods can be used in the recommendation model's starting phase. On the other hand, the content-based recommendation model performs interest pattern mining based on the characteristics of the user's interest entries, which essentially builds a model for calculating the degree of interest in an item, and the method does not depend on other users. However, the model is prone to the problem of duplicate recommendations and thus requires an additional step for de-duplication operations [20].

1.3 Research Objectives and Content

The purpose of this study is to propose a high school exercise recommendation system based on knowledge tracing, which is divided into three parts: the first part is the exercise knowledge point labeling part. This section requires text mining of math exercises, building a knowledge point labeling model, and performance testing of the model. In the second part, the labeled knowledge point exercises are adopted as the input of the knowledge tracing model, which learns the knowledge graph embedding representation of the exercises through graph neural networks, designs a knowledge tracing model based on the DKVMN model to track the students' knowledge mastery proficiency state and predict the correct probability of the next exercise, and outputs the hidden knowledge state vector, which is used in the later recommendation section as the next level input. In the last part, a recommendation model based on two stages of matching and ranking is designed first to filter the exercises to filter the candidate recommended exercises, then apply the knowledge tracing model to predict the probability of correct answers, and output a final list of recommended exercises.

1.4 Thesis Organization and Structure

Chapter 1 of this thesis is an introduction. The study's research background, the research progress, and review of related techniques and theories are presented. The research focus and objectives of this thesis are described. Based on the currently available models for requirements analysis, a solution based on a knowledge tracing exercise recommendation system is proposed and divided into three modules: exercise knowledge point labeling, knowledge tracing, and exercise recommendation.

Chapter 2 of this thesis focuses on the knowledge point labeling to the exercises. In the exercise resource recommendation system, it is necessary to analyze the exercises' knowledge points and then recommend exercises related to the knowledge points that students have insufficient mastery. The model is composed of two parts: exercise text information extraction and multi-label knowledge classifiers generator. In the experimental part, the model's effectiveness is verified by conducting performance comparison experiments with several baseline models.

Chapter 3 of this thesis proposed an improved knowledge tracing model for DKVMN, which is divided into parts such as student answering behavior characterization, exercise knowledge weight calculation, knowledge mastery calculation, and knowledge mastery modification. The thesis starts with a theoretical introduction of the design idea and related techniques and then verifies the validity of the model and evaluates the model performance by comparing it with the baseline model on a public dataset in the experimental part.

Chapter 4 of this thesis proposed a recommendation system model consists of two phases: matching and ranking. In the matching phase of the model, multiple candidate exercise subsets are generated separately by a multiplex-matching algorithm and then merged into the weighted ranking candidate exercises' final set. In the model's ranking phase, the candidate set exercises are used as the input to the knowledge tracing model, and then the the exercises that are predicted to be mistaken would be collected and prioritized to generate a list of recommended exercises.

Chapter 5 of this thesis presents the conclusions and future directions for improving each part of the model.

Chapter 2

Exercise Knowledge Point Mining Based on Graph Neural Network

2.1 Research Motivation

In this thesis's study topic, to establish a high-quality data source is the critical step of building the recommendation system. The construction of an exercise corpus is a complex and challenging task, requiring consideration of all aspects of the exercise data and adding enough additional information to subsequent data mining exercises. High school mathematics has hundreds of knowledge points, and on average, each point has dozens of exercises of different difficulty gradients from easy to difficult. The size of a high-quality math subject exercise corpus is 100,000 scale. In addition to the quantitative requirements, there are two fundamental issues in a quality exercise corpus, in addition to the quality of the questions and the matching of the educational content, one is the construction of the knowledge point relationship network, and the other is the construction of the knowledge point-exercise relationship.

Moreover, it is the optimization of these factors that leads to the construction of high-quality exercise corpus and the extremely high cost. The question databases on the market often lack attention to these factors, leading to the emergence of many poor-quality exercise databases. Therefore, labeling the questions' knowledge points and constructing the knowledge system is one of the most critical issues in building a quality exercise corpus.

One of the critical problems in building a test recommendation system is recommending topics in conjunction with knowledge points. When referring to the solution to this problem, the knowledge point labels should be considered. In textbooks and syllabi and various teaching aids, there are various descriptions of "knowledge points". Knowledge points in high school mathematics, such as functions, definition domains, value domains, or analytic equations, have direct concepts, applications of methods, abstractions of topics, and summaries of similar solutions. There is no one standard way to classify these, and the methods of knowledge point system construction may be very different. As far as the task of data mining is concerned, this thesis is more concerned with the concepts and skill points it involves, and these can be concluded for confidential information through the text of the exercises. As an exercise recommendation system that analyzes students' knowledge mastery proficiency, the knowledge point labels should be able to describe the core knowledge points, methods, or ideas of the topic quiz and be able to distinguish the ability requirement points for students to build a more robust user knowledge mastery model and recommendation engine.

2.2 Research Contribution

This chapter's main contribution is the relationships between knowledge points with graph networks' node information propagation mechanism, enabling the model to associate hidden related knowledge points through the tagged classified knowledge points. A multi-knowledge point label classification method for exercises is proposed, which uses the textual information described by the exercise text to identify the knowledge points related to the exercise, essentially a text classification task. The proposed method is divided into two parts: a bidirectional LSTM textual information representation network with attention mechanism and a multi-knowledge point labeled classifier generation network based on a graph convolutional network propagation mechanism. In the text information representation network, both the exercise and knowledge point texts are input vectors, and the output exercise text is used as the final exercise information representation. Subsequently, the knowledge point text vector is further learned by graph convolutional network propagation in the label generator network to embed the interrelationships among knowledge points. The final exercise representation vector is calculated by multiplying with the output knowledge point classification to obtain the exercises' knowledge point labels. After experimental validation, the proposed model outperforms the original DKVMN and other baseline models in proposed metrics.

2.3 Proposed Model

In this chapter, a multi-knowledge point labeling classification method for exercises is proposed. The method is divided into textual information characterization using Attention-based bidirectional LSTM and multi-knowledge point labeling classifier generation using the GCN embedding learning method. Finally, the classifier is used to perform the knowledge point labeling tasks on the exercises. The knowledge point labels are mined for topic recommendation and analysis reports. It requires refining the topic's knowledge point association, i.e., several corresponding knowledge point tags for the topic. There will also be dependencies between the knowledge points and build a complete knowledge map of knowledge points for generating student learning reports. The first problem is a knowledge point mining task using the topic text, which is also a classification task. To be more specific, it is a hierarchical classification task based on the topic's short text information. The most basic natural language processing (NLP) technologies and machine learning should be used in this task. By learning a large amount of manually labeled topic text and knowledge point labeling results (also called training corpus) - obtaining the features of the topic text through NLP techniques and obtaining the classification model through machine learning. The system can do knowledge point classification automatically. The object to be classified is a topic (including stem, answer, paraphrase, etc.), and the result is a set of knowledge point labels. The input to the learning system is a set of training exercises, i.e., *n* questions that have been labeled with their corresponding knowledge labels; the learning system trains the given classification model based on the training data. In the prediction phase, the exercises' input to be labeled is used to output a set of predicted labeling results for the exercises' knowledge points.

2.3.1 Algorithm Overview

This section aims to construct a model for mining the exercise-knowledge point relationship and mining the association between knowledge points. Establishing an exercise-knowledge point relationship means labeling the knowledge points of an exercise and collecting knowledge concepts to understand and solve the exercise. Therefore, accurately describing a test question's knowledge points is essential for the subsequent knowledge tracing and recommendation process. The two basic classification approaches that already exist are expert labeling and machine learning. The former means that education experts combine their professional knowledge to label knowledge points of test questions. However, when the number of questions or complexity of questions is high, manual labeling has a high workload, high subjectivity, and imperfect labeling. Also, considering the association between knowledge points, the manual labeling has the problem of not taking into account the inline knowledge relationship. Another way is to use the rule-based automated labeling method, which performs knowledge keyword matching by non-intelligent means such as text pattern matching. However, many exercises often do not have explicit knowledge point texts, so this correctness rate is not satisfactory. Also, an exercise often has multiple knowledge points, so in effect, knowledge point mining is a multi-label classification problem [21–23].

This chapter also discusses “how to effectively model the relationships between knowledge points” and proposes a multi-label classification model based on graph neural networks.

In 2019, a multi-label image classification model was proposed [24]. Inspired by this model, this chapter proposes a multi-knowledge point labeling model based on attention mechanism exercise description text feature extraction and graph convolutional neural network (GCN) based knowledge point relationship mining, which builds a knowledge point relationship graph to describe the association relationship between knowledge points by a data-driven approach, builds classifiers for knowledge points separately, and then performs multi-label classification. Its main architecture diagram is shown in Fig. 2.1.

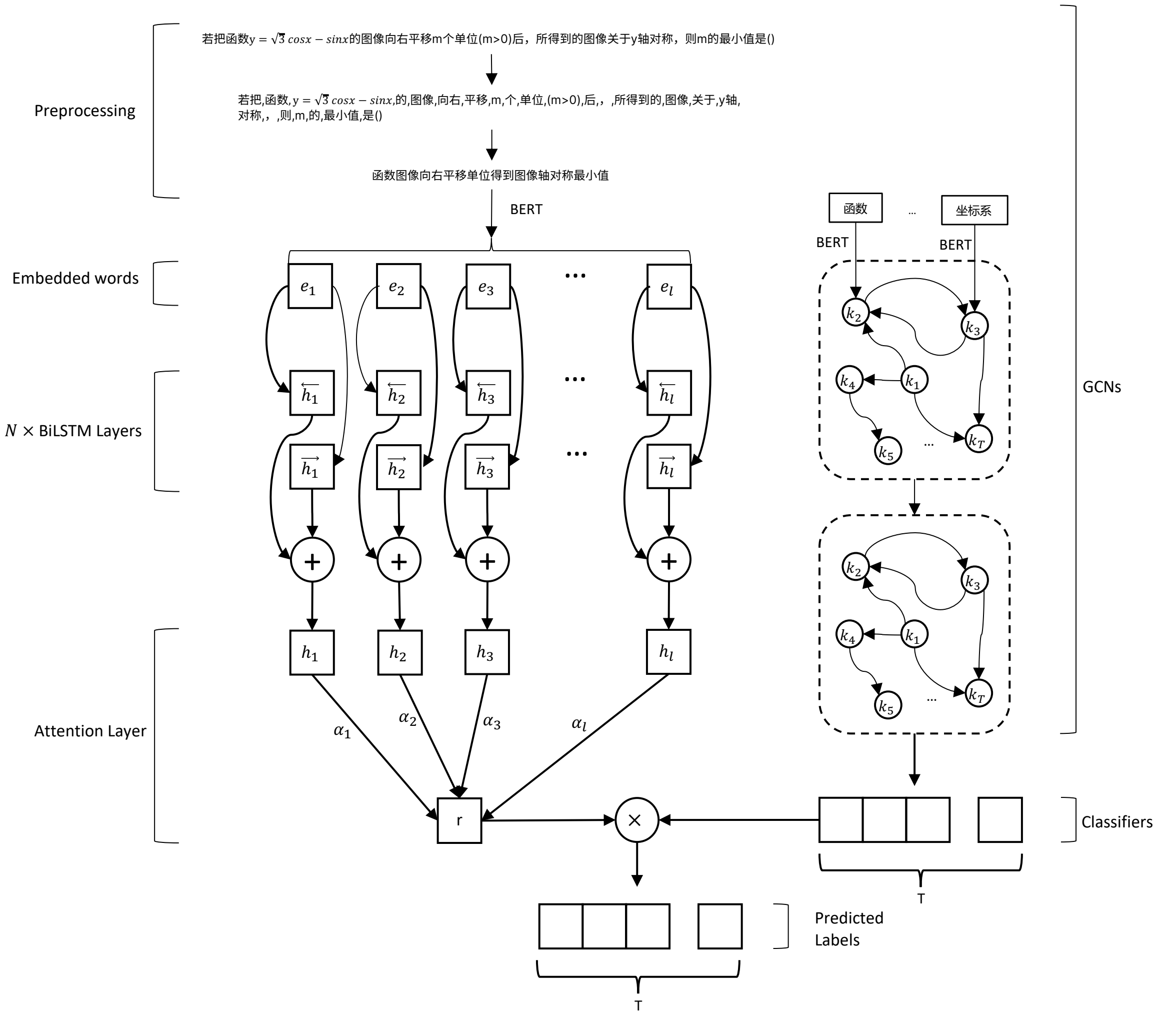

Preprocessing
若把函数y = √3 cosx − sinx的图像向右平移m个单位(m>0)后，所得到的图像关于y轴对称，则m的最小值是()
若把,函数,y = √3 cosx − sinx,的,图像,向右,平移,m,个,单位,(m>0),后,，,所得到的,图像,关于,y轴,对称,，,则,m,的,最小值,是()
函数图像向右平移单位得到图像轴对称最小值
BERT
函数
...
坐标系
BERT
BERT
Embedded words
e_1 e_2 e_3 ... e_l
N × BiLSTM Layers
$\overleftarrow{h_1}$ $\overleftarrow{h_2}$ $\overleftarrow{h_3}$ ... $\overleftarrow{h_l}$
$\overrightarrow{h_1}$ $\overrightarrow{h_2}$ $\overrightarrow{h_3}$ ... $\overrightarrow{h_l}$
+
h_1 h_2 h_3 h_l
Attention Layer
α_1 α_2 α_3 α_l
r
×
k_1 k_2 k_3 k_4 k_5 ... k_T
GCNs
Classifiers
T
Predicted Labels
T

Fig. 2.1 Structure of the knowledge labeling model

From the structure, it can be seen that there are generally two parts to the model:

1. The first part is the exercise description text information mining module, which uses a Bi-LSTM network with an added attention mechanism to text-mine hidden knowledge information on questions (including question descriptions and answers). In this thesis, an end-to-end network training approach is designed to achieve the model's overall iterative learning. Specifically, it consists of a text preprocessing part that performs the text subdivision, filtering, and deduplication parts, an embedding layer that performs the word vector embedding calculation, and an Attention-based Bi-LSTM (Bi-LSTM) network that performs text information mining, which was proposed by Peng et al. in 2016 [25], outputting a textual information representation vector.

2. The second part is a GCN-based knowledge point association multi-label classifier, which maps knowledge points to a set of interdependent target classifiers. These classifiers are multiplied with the exercise information representation vector outputted by the first part to output a result vector representing each knowledge point's labeling probability.

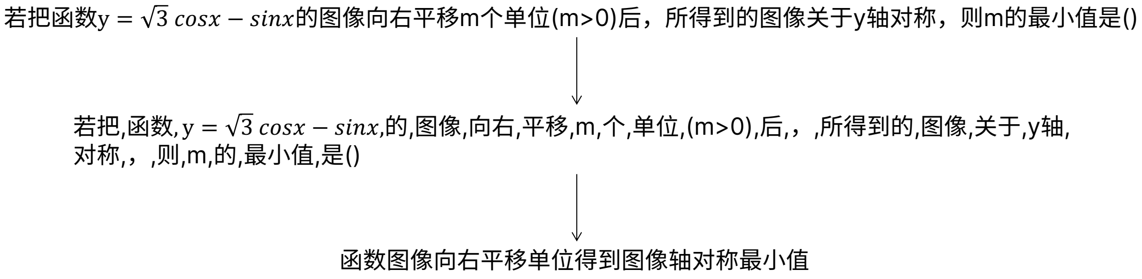
2.3.2 The Exercise Description Text Mining

This section is based on the attention-based Bi-LSTM, including four layers: Preprocess Layer, Embedding Layer, Bi-LSTM layer, Attention Layer, and Output Layer:

Pre-process Layer

The preprocessing stage mainly includes word separation, cleaning, and regularization. Considering that our research object is Chinese high school mathematics test questions, compared with English, there is no middle space in the middle of sentences of Chinese language, so it is necessary to use the word separation algorithm to decompose the sentences into subwords. There are many texts in the content that are irrelevant to the sentence expression, which will cause much interference and redundant information if the calculation is performed directly, so additional text cleaning is also a necessary step. The Fig. 2.2 shows an example of preprocessing of an exercise description text.

- Segmentation: Since original data has many ambiguous words and phrases, a word segmentation step is required to remove irrelevant words and phrases. In this step, the model removes irrelevant punctuation, stops words and words from the user dictionary. The user dictionary contains mainly redundant words from the text of mathematical exercises.



若把函数$y = \sqrt{3}\ cosx - sinx$的图像向右平移m个单位(m>0)后，所得到的图像关于y轴对称，则m的最小值是()

若把,函数,$y = \sqrt{3}\ cosx - sinx$,的,图像,向右,平移,m,个,单位,(m>0),后,，,所得到的,图像,关于,y轴,对称,，,则,m,的,最小值,是()

函数图像向右平移单位得到图像轴对称最小值

Fig. 2.2 Example of preprocessing

- Cleaning: Corpus cleaning preserves useful data in the corpus and deletes noisy data. Common cleaning methods include manual deduplication, alignment, deletion, and labeling. For words that are not necessary for a sentence, i.e., stop words, their existence does not affect the sentence's meaning. There will be many function words, pronouns, or verbs, and nouns with no specific meaning in the text. These words are not helpful to the text analysis so that these stop words can be removed. For the exercise description text, there are many mathematical expressions and symbols. Considering that these expressions in many exercises are not in text format, OCR technology must be applied to preprocess mathematical expressions from pictures to text. Therefore, when the Chinese text is sufficient, mathematical expressions can be removed to reduce the calculation load.

After the data processing step, a clean sequence of text tokens is obtained, and next in the Embedding layer, the BERT technique can be used to perform text embedding operations.

BERT-Based Embedding Layer

In applying deep learning, embedding as a preprocessing approach for generating embedded vectors brings a great extension to neural networks' application in various aspects. In applying deep learning techniques, embedding is an instrumental skill because it reduces the spatial dimensionality of a discrete variable and allows a meaningful representation of that variable. For example, in NLP, if basic one-hot coding is used, it often results in invalid dimension and sparse vectors and also fails to learn the dependencies between vectors. The embedded vectors are updated during embedding training, which can clearly show the exercises between the vectors. The one-hot encoding is the most naive method, which turns all words into binary patterns, i.e., all words are only present or absent in two cases, so each word is a binary vector with only 1-value and all other 0-values. When the amount of words is large, this vector's length will also be quite long, and the subsequent computation will also generate a large number of invalid computations, i.e., the sparse matrix computation problem. Word2vec was proposed by Google Language 2013 [26], which predicts its context by words or predicts words by contexts, a static word embedding learning model, but encounters bottlenecks in solving problems such as polysemous words. The Bidirectional Encoder Representations from Transformers (BERT) model proposed by Google in 2018 utilizes the Transformer as the base unit to pre-train masked language models, achieving State of The Art (SOTA) performance in almost all NLP tasks. It has a powerful semantic representation effect. In this thesis, BERT is utilized as a word embedding vector learning module to achieve greater generalization and adaptive capabilities in different contexts of different idiomatic texts.

In the exercise description text, there will be some combinations of words, i.e., groups of words with indivisible lexical meanings formed by combining multiple words. Since the words are masked during the BERT training sample generation phase, these combinations may be masked separately, resulting in ambiguity and causing training performance degradation. Therefore, this thesis applies Whole Word Mask processing (WWM), proposed by Cui et al. in 2019 [27]. By applying WWM, when one part of a combined word is masked, then the other parts of that combined word are also masked, as shown in Fig. 2.3.

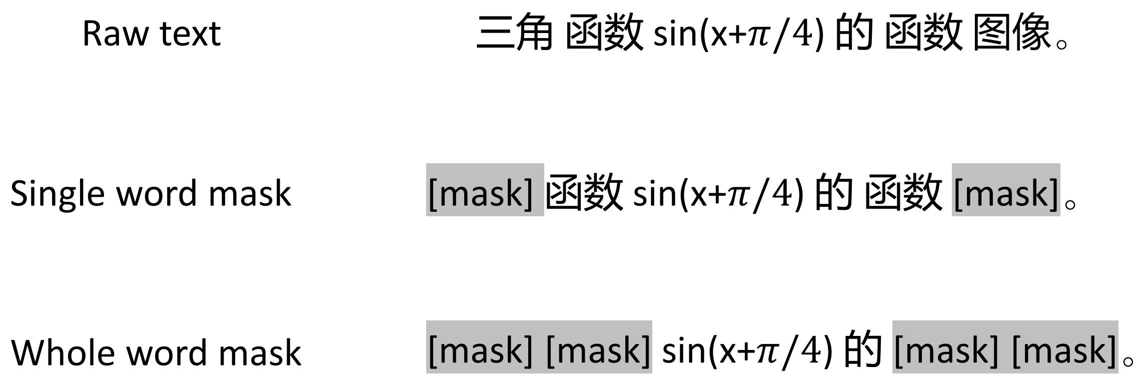

Raw text
三角 函数 sin(x+π/4) 的 函数 图像。
Single word mask
[mask] 函数 sin(x+π/4) 的 函数 [mask]。
Whole word mask
[mask] [mask] sin(x+π/4) 的 [mask] [mask]。

Fig. 2.3 Whole word mask

WWM processes some words of the original sequence of exercise description text words obtained after the word separation process, and the marker [CLS] is added at the beginning of the sequence, and the inter-sentence is marked by [SEP] separator. After training, the word embedding vector is output. Each word's output embedding consists of three parts: token embedding, segment embedding, and position embedding, which characterize the word's embedding information from different perspectives. Subsequently, the word embedding vector is fed into the feature extraction bidirectional Transformer layer of BERT, and the feature sequence representation vector containing deep semantic features can be obtained. The overall architecture is shown in the Fig. 2.4.

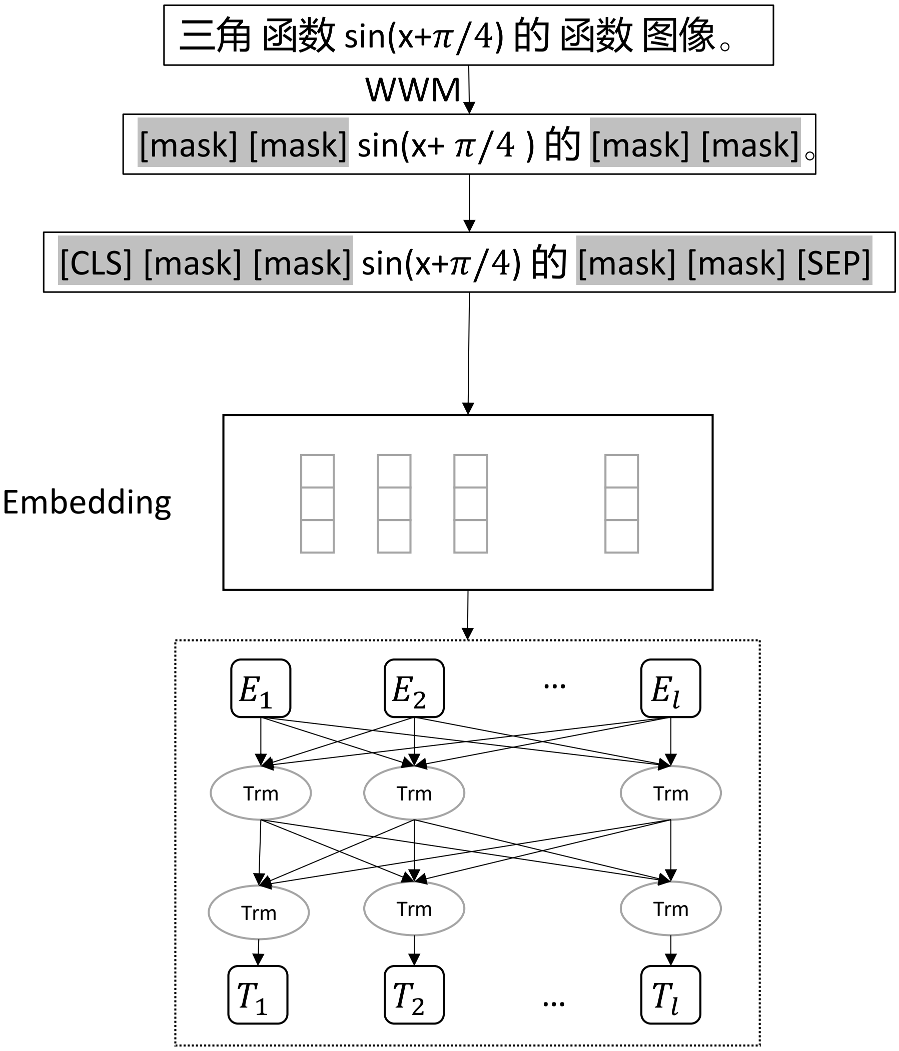

三角 函数 sin(x+π/4) 的 函数 图像。
WWM
[mask] [mask] sin(x+ π/4) 的 [mask] [mask]。
[CLS] [mask] [mask] sin(x+π/4) 的 [mask] [mask] [SEP]
Embedding
E1
E2
...
El
Trm
Trm
Trm
Trm
Trm
Trm
T1
T2
...
Tl

Fig. 2.4 BERT-based embedding

Bidirectional LSTM Layer

RNNs are well equipped to solve various types of problems and tasks in serialized pattern data modeling. In this section, the core of the task is a sequence-to-sequence (Seq2Seq) task, where an embedding vector described by the topic is input and a sequence containing the currently trained information is output. The most rudimentary idea is to use the original RNN. Its structure is shown in the Fig. 2.5, where x_t, h_t and y_t denote the input, hidden and output values at time t, respectively. Then, the RNN training formula can be written as Eq. 2.1:

$$\begin{aligned} h^{t+1} &= f(W_t^h h^t + W^i x^t) \\ y^{t+1} &= f(W^o h^{t+1}) \end{aligned} \tag{2.1}$$

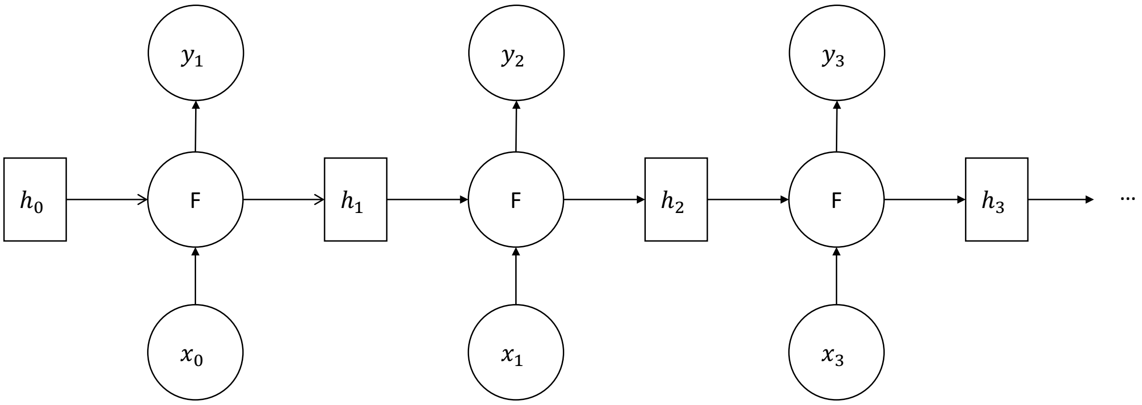


Fig. 2.5 Naive RNN unit

However, when the sequence is long, the problem of long dependencies arises. For example, for a sequence, the current state depends on a state far away from the current state, and as the time interval increases, the ability of RNNs to learn state representation is greatly reduced. Long short-term memory (LSTM) [28, 29], as a solution to overcome shortcomings of RNN, which uses a gating mechanism to achieve long-term memory. It solves the problems such as gradient explosion or gradient disappearance of RNN. It also has a good performance in capturing sequence information. The general model of LSTM is like Fig. 2.6, which represents the computational details within an LSTM cell at the moment of t. Among them, σ is the activation function, c_t is the cell state representation, f_t is the forgetting gating calculation, i_t is the input gating calculation, o_t is the output gating calculation, and h_t is the hidden state representation. The forgetting mechanism controlled by gating can be effectively modeled for long-range sequential unitary information.

One-way LSTM also has an inherent drawback that it can only capture the sequence state information before t moment, i.e., it can only capture the previous sequence input. The bidirectional LSTM (Bi-LSTM) can capture semantic information in both directions by feeding the reverse sequence into the LSTM and aggregating it with the forward LSTM sequence while modeling the dependencies in both directions [30]. The Bi-LSTM output can be obtained by inputting the positive sequence and the reverse sequence input sequence into

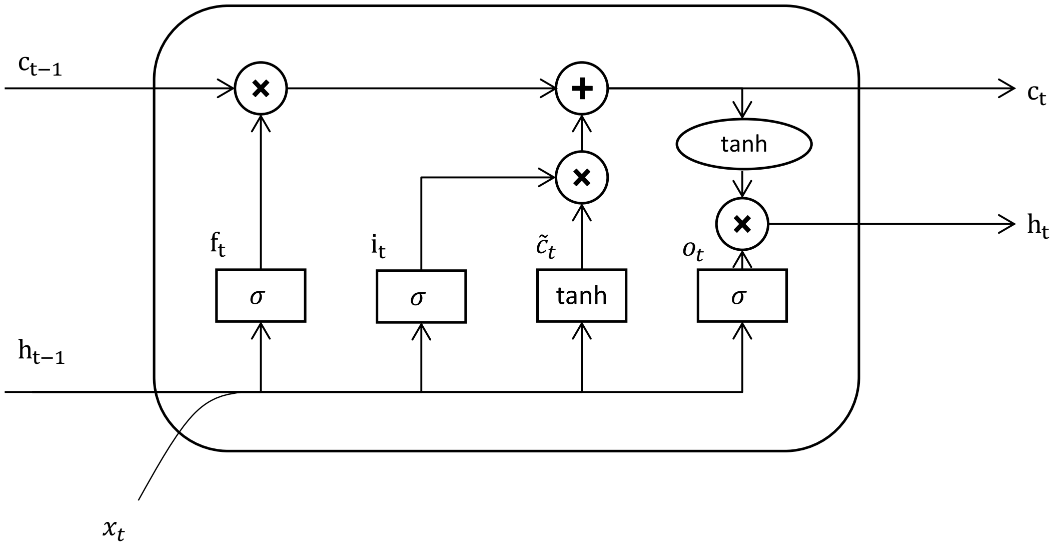


Fig. 2.6 Structure of LSTM unit

two sets of LSTM networks and perform element-wise addition. The output of positive-order LSTM is $\overrightarrow{h_t}$, the output of reverse-order LSTM is $\overleftarrow{h_t}$, $\bigoplus$ means sequence concatenation. The output of Bi-LSTM is shown in Fig. 2.2.

$$\begin{aligned}\overrightarrow{h_t} &= \overrightarrow{LSTM}(e_t)\\ \overleftarrow{h_t} &= \overleftarrow{LSTM}(e_t)\\ h_t &= \overrightarrow{h_t}\bigoplus\overleftarrow{h_t}\end{aligned} \tag{2.2}$$

The Bi-LSTM Structure is like Fig. 2.7.

Here, the Bi-LSTM output a bidirectional sequence as the concatenation of positive-order and reverse-order LSTM output sequence. It is a higher-order representation of the exercise text's sentence vector, and the model can also be stackable with multiple Bi-LSTM layers to characterize the deep sentence sense vector [31].

Attention Layer

Since each feature word's impact on the overall semantics in a text mining task is asymmetric, i.e. keywords have a decisive role in determining the meaning of the entire text [32]. Human attention is a mechanism that focuses on key information ignoring non-key information, i.e., individual information points are weighted differently. This method achieved

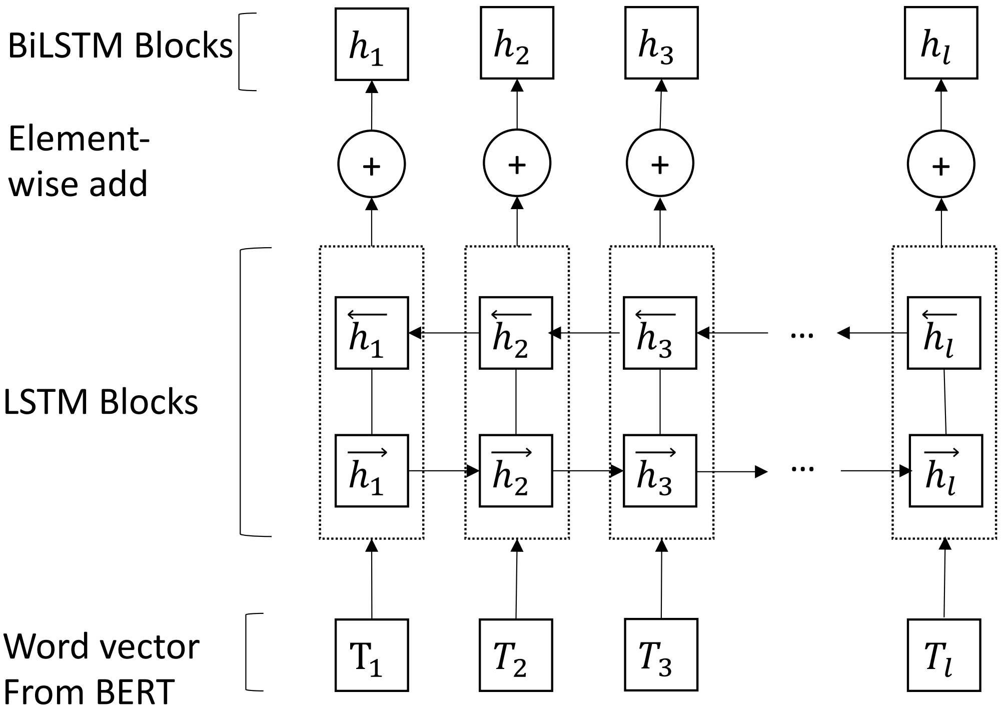


Fig. 2.7 Structure of Bi-LSTM

remarkable results in different tasks such as image vision [33, 34], language mining [35], and voice recognition [36] and is applied widely.

Human attention is a mechanism for quickly screening high-value information from massive information. The human attention mechanism inspires the deep learning attention mechanism. This method is widely used in various types of deep learning tasks such as NLP [35], image classification [33, 34], and speech recognition [36], and has achieved remarkable results.

The essence of the attention mechanism is a group of key-value pairs $< K, V >$ contained in Source S, given an element of Query Q, which is followed by calculating the Q similarity to each K and remembering the weight coefficients of the corresponding values V. It is showed in Fig. 2.8. Then a weighted sum is performed to obtain the final Attention value. It can be expressed as Eq. 2.3, where Att and Sim represent attention and similarity.

$$Att(Q, S) = \sum_{i=1}^{|S|} Sim(Q, K_i) * V_i \tag{2.3}$$

Following the attention method, the encoder use Bi-LSTM and get hidden state vector $h_t = \overrightarrow{h_t} \bigoplus \overleftarrow{h_t}$. A word attention mechanism is introduced here. The core principle of the

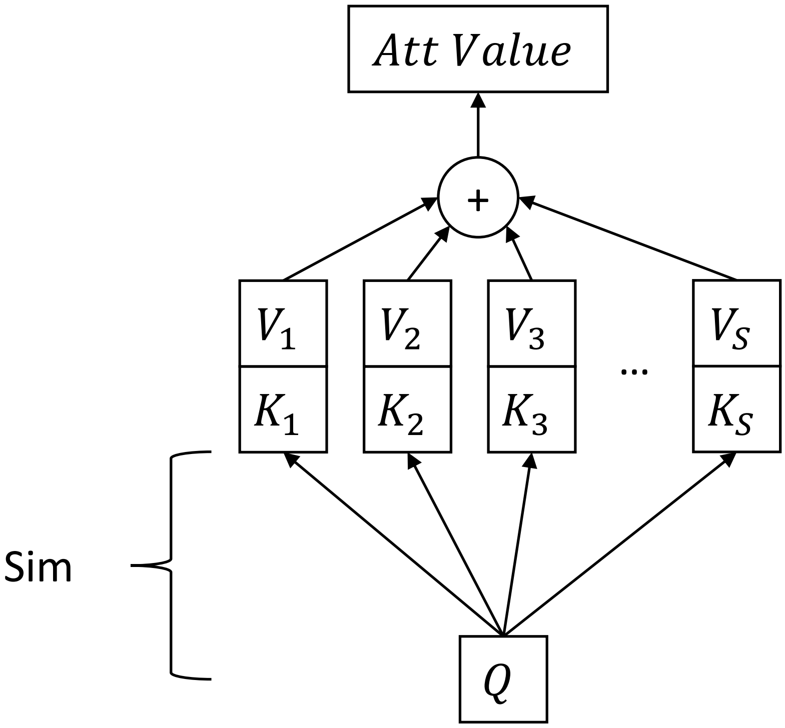


Fig. 2.8 The essential idea of attention mechanism

attention mechanism in this section is to compute the words that have the greatest influence on sentence meaning, i.e., the key semantic words, which are later aggregated into a vector of sentence meaning representations. The formula is Eq. 2.4, where u_t is the hidden representation of h_t, r is the output text vector, and the u_ω is the similarity of the text vector of word aspect. By measuring the similarity between u_t and u_ω as the importance of the word, and using the softmax function to calculate normalization and to obtain the importance weight α_t. Finally, the entire text is transformed to word representation based on semantic contribution weights.

After that, the entire text is represented as a weighted sum of word vectors. The context vector u_ω can be regarded as a high-level representation of the fixed query "what is the word conveying information" and is randomly initialized as a learnable parameter in training.

$$
\begin{aligned}
u_t &= \tanh(W_\omega h_t + b_\omega) \\
\alpha_t &= Softmax(u_t^T u_\omega) = \frac{\exp(u_t^T u_\omega)}{\sum_t \exp(u_t^T u_\omega)} \\
r &= \sum_t \alpha_t h_t
\end{aligned}
\tag{2.4}
$$

2.3.3 The GCN-based Knowledge Point Classifier Generator

In high school mathematics, knowledge points have more complex interrelationships such as correlation, subordination, inclusion, predecessor, successor, etc. These complex interrelationships are often difficult to model in Euclidean space, or this creates data sparsity problems. The establishment of relationships between knowledge points through graph data structures is more intuitive and has better interpretability. Recalling this model's task, it gives descriptions and answers to exercises, some of which have already marked knowledge points, and uses this information to mark knowledge points for exercises that are labeled knowledge points. Considering the dependence between knowledge points, some deeply hidden knowledge points that cannot be represented in shallow features can also be correctly labeled by the graph neural network output classifier. In this model, a GCN is used to learn and form the knowledge point connection graph, and each knowledge point corresponds to a node on the graph. After multiple graph convolution calculations, a series of classifiers are generated. These classifiers respectively act on the text vector generated by the text mining module. Each classifier outputs a value representing the probability that the knowledge point is associated with the exercise. Its overall structure is shown in the Fig. 2.9.

Graph Convolutinal Network

In the real world, many important data sets generate connections as networks that form graph-like structures. Several research reviewed this problem and attempted to generalize neural networks and apply them to arbitrary graph-structured data [37, 38]. The GCN [7] is used to process non-Euclidean spatial data that are difficult to learn by traditional convolutional neural networks. It is defined on a graph structure $G = (V, E)$, where V is the denote of vertex and E is the denote of edge inside the graph. The input of GCN $X = H^{(0)}$ is the first layer of GCN. The ith layer is denoted as $H^{(i)}$, and $H^{(i)} \in \mathbb{R}^{N^{(v)} \times d_i}$, where $N^{(v)}$ is the denote of number of vertexes and d_i is the number of dimension of each vertex of layer i. The output layer $Z = H^{(L)}$ where L is the number of layers in GCN. Also, the GCN has another adjacency matrix A of dimension $N^{(v)} \times N^{(v)}$ used to describe the graph structure.

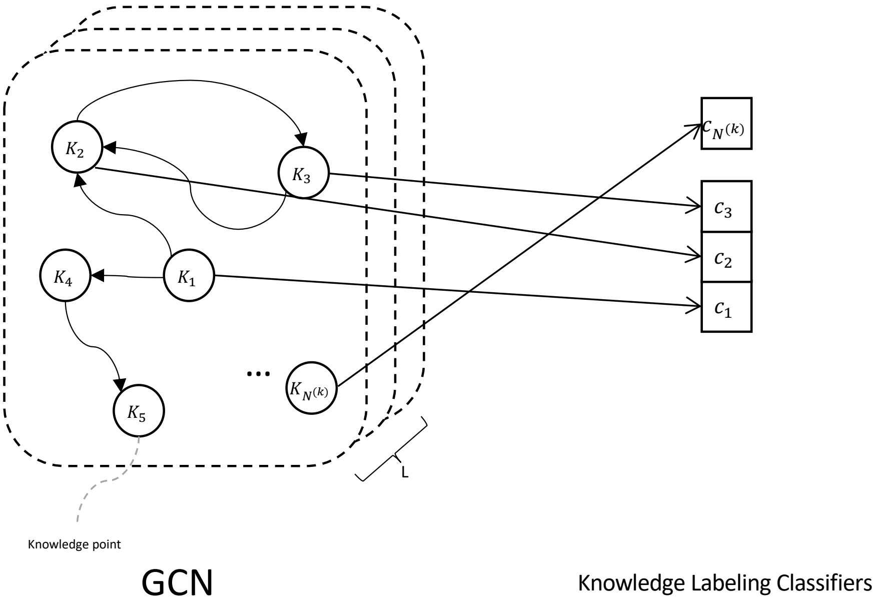


Fig. 2.9 The overview of structure

The core of graph convolutional learning is propagation, i.e., each node propagates information to neighboring nodes. The propagation of each layer is aggregated to form the next layer. The $i - 1$ layer $H^{(i)} \in \mathbb{R}^{N^{(v)} \times d_v}$ to layer $H^{(i+1)}$ transformation of GCN can be written in the formula Eq. 2.5, where $f(\cdot)$ is a specific propagation method, e.g. all nodes spread their own values uniformly to neighboring nodes.

$$H^{(i+1)} = f(H^{(i)}, A) \tag{2.5}$$

A correlation matrix can represent the relationship between knowledge points, i.e., when a knowledge point i is related to a knowledge point j, the correlation matrix A models the knowledge point as a vertex in the GCN in order to learn the representation between knowledge points. The relation is learned by co-occurrence probability, i.e., supervised learning is performed on the library of exercises that have been tagged with knowledge points, which is based on the assumption that when multiple knowledge points have a high probability of occurring in an exercise, they should be intrinsically linked.

The node propagation of GCN in this section is Eq. 2.6, where $\widehat{A}$ is the normalization form of $A \in \mathbb{R}^{N^{(v)} \times N^{(v)}}$. The $h(\cdot)$ is the nonlinear activation function LeakyReLU [39]. The parameter matrix $W^{(i)} \in \mathbb{R}^{d_i \times d_{i+1}}$ to be learned can be calculated by the statistical relations of the exercise knowledge points. The training process of Proposed is shown in Fig. 2.10.

$$H^{(i+1)} = f(\tilde{A}H^{(i)}W^{(i)}) \tag{2.6}$$

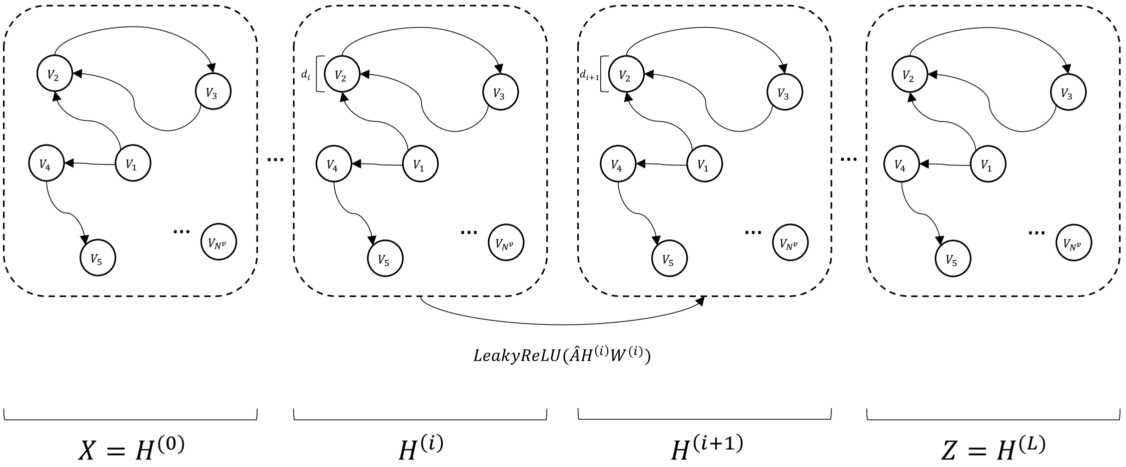


Fig. 2.10 The GCN training process

In this model, each node represents a knowledge point. The graph input is the word embedding of the label. As the model integrates the text information of labels, so each node has latent relation with exercises represented with a text representation vector, which is the underlying principle of this model. The graph propagation learning of related knowledge points can produce the fusion of related knowledge point representations, which improves the associative learning performance of model recognition for related knowledge points.

Design of Correlation Matrix

In GCN, the correlation matrix A characterizes the relationship between graph nodes, and GCN information propagation calculation is also based on A. Therefore, designing the correlation matrix A is a crucial step in the GCN model. In this model, the commonly used data association rule mining algorithm Apriori algorithm calculates knowledge point association by knowledge point reference co-occurrence statistics.

In this model, only the knowledge pairwise relationship needs to be found. Therefore, the knowledge point relationship matrix can be expressed as $R \in \mathbb{R}^{T \times T}$. The first task is to find the number of occurrences of frequently associated label in the label set. The Support, Confidence, and Lift can be used to evaluate frequent label sets. Support is the proportion of the number of occurrences of label pair in the label set in the total label set. Confidence degree reflects the probability of a label L_i appearing, another label L_j appears, or the conditional probability of the data. Lift represents the probability that the label L_i is contained at the same time, and the ratio of the probability of occurrence of X population:

$$\begin{aligned}
\text{Support}(L_i, L_j) &= P(L_i, L_j) = \frac{\text{number}(L_i, L_j)}{\text{number(All Samples)}} \\
\text{Confidence}(L_i \rightarrow L_j) &= P(L_i \mid L_j) = P(L_i, L_j)/P(L_j) \\
\text{Lift}(L_i \rightarrow L_j) &= P(L_i \mid L_j)/P(L_i) = \text{Confidence}(L_i \rightarrow L_j)/P(L_i)
\end{aligned} \tag{2.7}$$

Similar to calculating Support, the frequency matrix $E \in \mathbb{R}^{N^{(k)} \times N^{(k)}}$ of the sample knowledge point label pairs in the exercise training set can be calculated here. The M_{ij} represents the amount of co-occurrence between the knowledge point i and the knowledge point j in an exercise reference. Similarly, the knowledge point pair can be calculated by calculating Confidence to calculate the conditional probability matrix P, where $P_{ij} = P(L_i, L_j)/P(L_j)$, where $P_{ij} = P(L_i, L_j)/P(L_j)$ means the situation when the knowledge point j appears The conditional probability of the occurrence of the following knowledge point i.

It is a simple solution to directly set P as the incidence matrix A, but in actual situations, some comprehensive questions in the exercise set contain practically unrelated knowledge points, but these situations are relatively rare. In order to exclude the interference of accidental circumstances, a minimum knowledge confidence threshold can be set. When P_{ij} is greater than the given threshold $\tau^{(k)}$, naming activating value, then A_{ij} is set to P_{ij}, Otherwise A_{ij} is set to 0. The formula is like Eq. 2.8.

$$A_{ij} = \{ \begin{array}{ll} 0, & \text{if } P_{ij} < \tau^{(k)} \\ P_{ij}, & \text{if } P_{ij} \geq \tau^{(k)} \end{array} \tag{2.8}$$

This model adopts the GCN structure because the parameter sharing of GCN for each node allows the learned classifier to retain the associated information in the knowledge association graph, thereby implicitly expressing its spatial semantic structure. Therefore, the output classifier can retain and identify the implicit knowledge label dependent information. For the dependence of knowledge points, refer to the data association method mining algorithm such as Apriori algorithm [40], and calculate the correlation matrix by calculating the co-occurrence of knowledge points in the exercises.

Classifier generator

In this thesis, a learnable stacked GCN-based model for knowledge relation representation is proposed. The knowledge point labels are represented by knowledge label word embedding. Knowledge point set $K = \{k_1, k_2, \dots, k_{N^{(k)}}\}$, where $N^{(k)}$ denotes the amount of knowledge points. For each single knowledge point k_i, it is constrained that $k_i \in \mathbb{R}^{d^{(k)}}$, in which $d^{(k)}$ is the dimensionality of the embedding vector of knowledge point object. The knowledge label word embedding set is the input of GCN, i.e., $X = K$. The GCN is used to transform these knowledge point objects one by one into an internally connected knowledge point object classifier $C = [c_1, c_2, \dots, c_{N^{(k)}}$, where $c_i \in \mathbb{R}^{d^{(r)}}$, $d^{(r)}$ is the dimensionality of the text representation vector r output by the text mining module. These classifiers and r can be used to calculate the dot product of each label. The structure overview is proposed in Fig. 2.11.

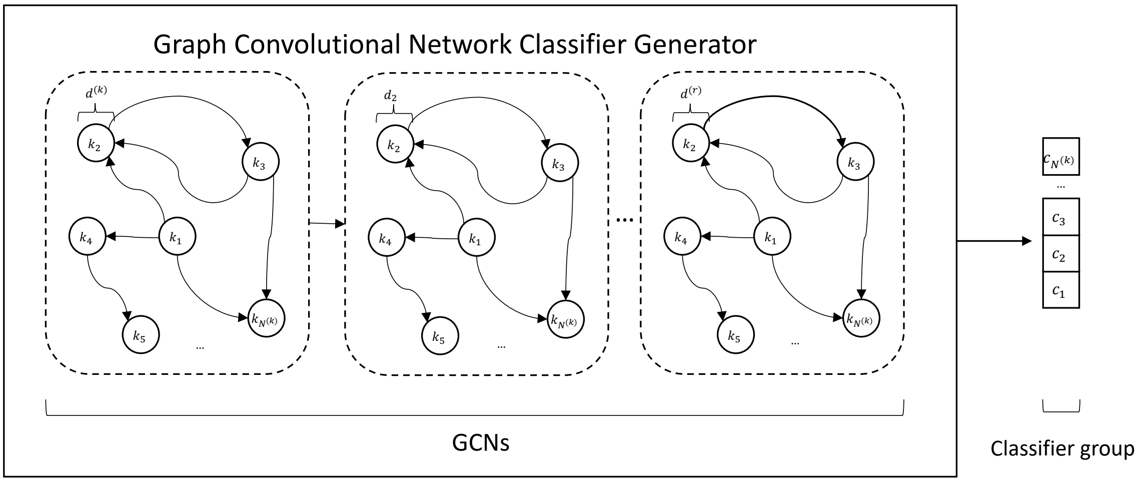


Fig. 2.11 The structure of GCN-based classifier generator

2.3.4 Multi-label Recognition

From the stacked GCN network, the object classifiers $C = \{c_1, c_2, \dots, c_{N^{(k)}}\}$ can be learned, where $N^{(k)}$ represents the number of knowledge points. Finally, the label prediction vector $\hat{y}$ can be obtained by the dot product of the learned classifier and the title text representation r as shown in Eq. 2.9.

$$\hat{y} = C \times r \tag{2.9}$$

Through manual labeling, the real knowledge point labels of the exercises can be obtained: $y \in \mathbb{R}^C$, $y_i \in \{0, 1\}$, $y_i = 0$ means that the exercise does not have knowledge points The label of i, on the contrary, $y_i = 1$ means that the exercise has a label of knowledge point

i. The loss function **L** can be written as Eq. 2.10.

$$\mathbf{L} = \sum_{i=1}^{T} y_i \log(\mathrm{sigmoid}(\hat{y}_i)) + (1 - y_i) \log(1 - \mathrm{sigmoid}(\hat{y}_i)) \tag{2.10}$$

2.4 Experiments

This section first introduces, preprocess, and analysis the dataset, then introduces some baseline performance models, followed by the introduction of multi-label labeling metrics. Finally, comparison results and analysis are given.

2.4.1 Dataset

The experimental data are the labeled math examination questions and practicing exercises (including answer analysis) fetched from the online exercise website "tiku.21cnjy.com". The text corpus, such as exercise texts and answer analysis, are crawled through web crawlers. The raw data are roughly classified text records that contain a lot of repetitive redundancy and error information, so a data preprocessing process is needed to filter out the appropriate data. The knowledge graph of the original dataset is shown in the Fig. 2.12.

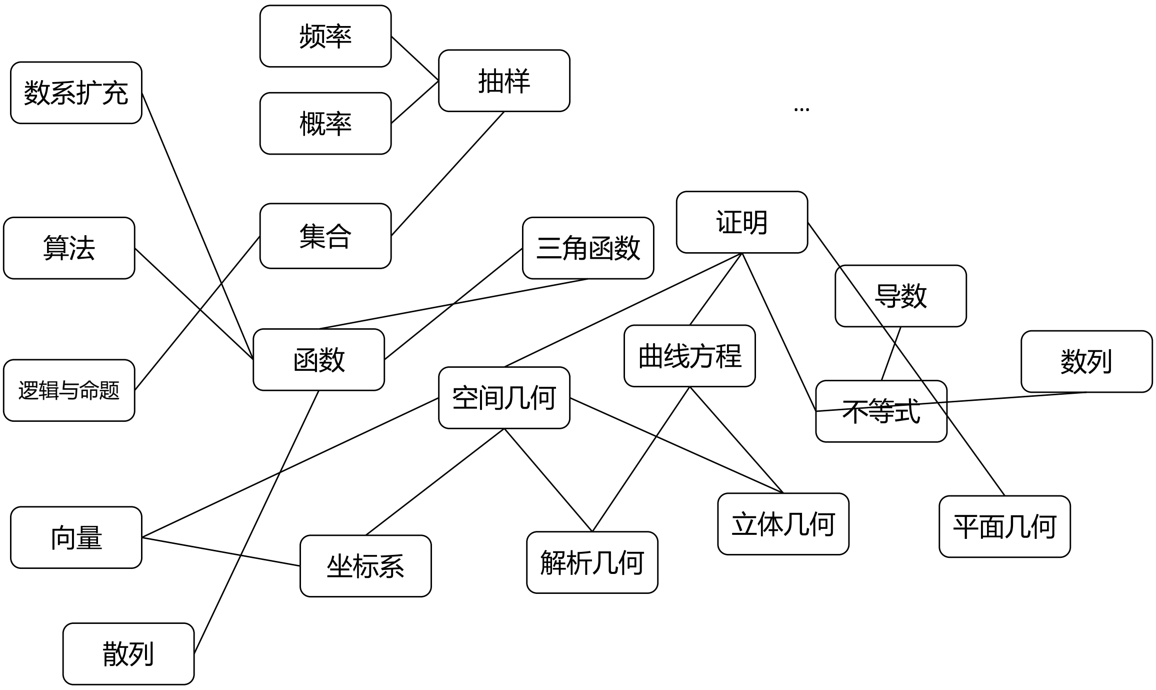


Fig. 2.12 Knowledge point relation graph of original dataset

Data Preprocess

This section uses web crawlers to obtain the data, so the original data obtained contains many unstructured exercises with irregular mathematical symbols, which will bring errors so that the unrecognizable characters will be replaced in the preprocessing process. There is also a large amount of nonsensical text. Besides, since some of the exercises are not fully labeled with knowledge points, manual assistance is required to label the dataset's training knowledge points.

Data Analysis

Since there are many images in the original set of exercises that cannot be identified, this question screens out irrelevant image URL information. This model does not consider textual information such as formulas and mathematical symbols, so only Chinese text will be screened out. Samples with relatively low text length will be removed.

In this experiment, a few of the more representative categories were selected to construct the dataset, and a total of 7 categories of knowledge points were selected, within which two categories are correlating. After deduplication and removal of invalid data, 3969 pieces of data were finally obtained. The filtered data are shown in the Table 2.1 and the distribution of the number of dependent exercises for each category is shown in the Table 2.2 and Fig. 2.13. The distribution of length of exercise texts is shown in Fig. 2.14. The results show that most samples' text lengths are concentrated in the interval range of 20-100, so values in this range can be considered when choosing the model's sequence length parameter. Also, the category distribution has a certain degree of non-average-ness. Classification metrics such as the F1 score should be used to exclude such effects.

Table 2.1 The filtered data examples

Exercise text	Knowledge labels
设偶函数的定义域为当时是增函…	函数奇偶性/导数
设奇函数的定义域为若当时的图…	函数奇偶性
设全集是实数集函数的定义域为…	集合
命题的逆命题是…	逻辑与命题关系
…	…

In addition, for each category, the inter-correlation between the categories can be analyzed and a heat map can be drawn as shown in Fig. 2.15. The results show that some categories in the processed dataset have direct positive correlations and others have negative correlations.

Table 2.2 The distribution of exercise in each category

Label	Number of exercises
三角函数	71
函数奇偶性	520
导数	750
平面向量	618
数列	746
逻辑与命题关系	526
集合	155

2.4.2 Baseline

This section's study focuses on a multi-label text classification task, so several more popular text classification models are used as baseline models. Some existing text classification models are selected experimentally to compare with the proposed model, and the performance is compared with the proposed model on the selected experimental evaluation metrics and the same dataset. At present, there are already some algorithm models for labeling text. To verify the effectiveness of the algorithm proposed in this thesis, the following baseline models will be set as comparisons.

- Bi-LSTM+Attention [25]: the model consists of an Embedding layer, a bi-directional LSTM layer, and an Attention layer. The text is passed through the Embedding layer to generate the Embedding vector, the LSTM to compute the higher-order sentence vector, the Attention layer to weight the sentence vector, and the final output layer to output the predicted labels.

- fastText [41]: Facebook proposes the fastText model in 2017. The fastText accelerated training and can be applied to text classification tasks, and it contains input, hidden, and output layers. The method is actually a weighted sum of the word vectors of the sentences as sentence vectors, using softmax classification.

- TextCNN [42]: Kim et al. adapted the CNN to the text classification task by improving it in 2014 [43]. It contains parts such as the convolutional layer, max-pooling layer, and output layer. The principle is to extract higher-order features of text by multiple convolutional kernels, and it can introduce pre-trained word vectors to improve the performance.

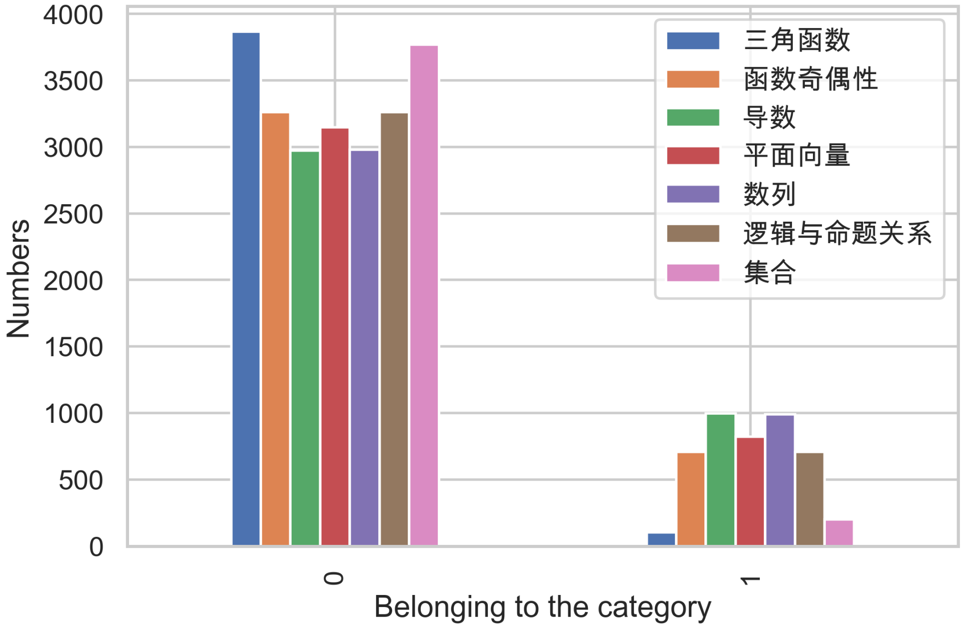


Fig. 2.13 The distribution of exercise in each category

2.4.3 Metrics

Compared with common text classification problems, it is hard to label multi-label data, and multi-label means huge classification cost [22]. In the tasks in this section, the test questions need to be labeled with multiple knowledge points. It is a multi-label classification problem. The sample dimension, data volume, and label dimension will all affect the labeling effect. This section uses label-based metrics to evaluate the performance of the proposed test knowledge point labeling model.

Regarding multi-label classification, considering the relationship between the exercises' labels, this thesis uses label-based indicators for model evaluation. The indicator calculates the confusion matrix indicator for each label to calculate the sample values of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). Then calculate the macro-F1 and micro-F1 metrics for multi-label tagging model performance evaluation.

According to the metrics proposed by Zhou et al. [22], for the jth label of the ith example in the n examples, shown in the Eq. 2.11, where x_i and L_j represents the ith exercise and jth label, Y_i and $\tilde{Y}_i$ represent the real labels and predicted labels of the ith exercise.

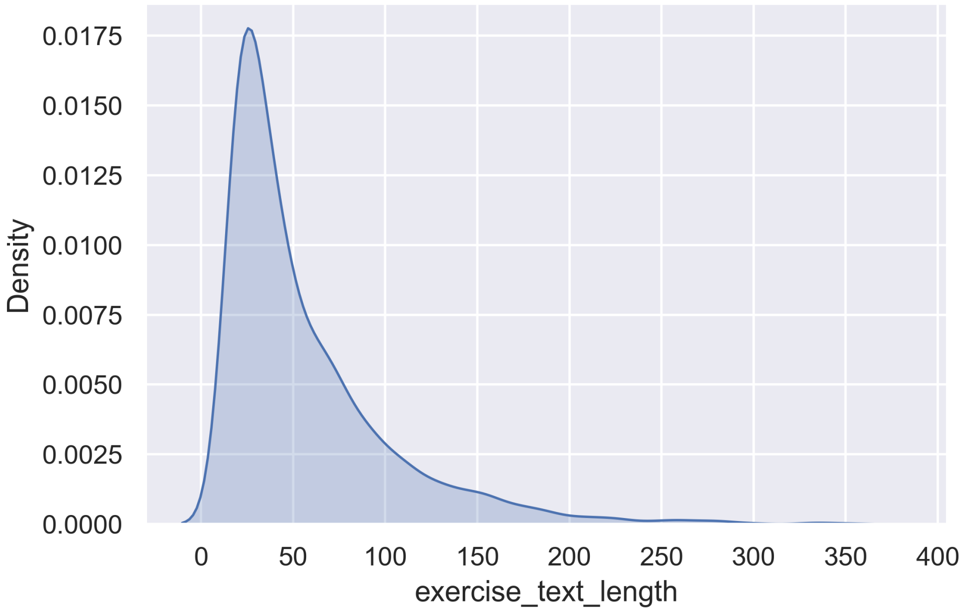


Fig. 2.14 The distribution of exercise text length

$$
\begin{aligned}
\mathrm{TP}_j &= |\{x_i | L_j \in Y_i \wedge L_j \in \tilde{Y}_i, 1 \le i \le n\}| \\
\mathrm{FP}_j &= |\{x_i | L_j \notin Y_i \wedge L_j \in \tilde{Y}_i, 1 \le i \le n\}| \\
\mathrm{TN}_j &= |\{x_i | L_j \notin Y_i \wedge L_j \notin \tilde{Y}_i, 1 \le i \le n\}| \\
\mathrm{FN}_j &= |\{x_i | L_j \in Y_i \wedge L_j \notin \tilde{Y}_i, 1 \le i \le n\}|
\end{aligned}
\tag{2.11}
$$

In the confusion matrix, Precision P is the proportion of data with correct predictions to the data that is predicted to be positive, and Recall R is the proportions of data with predictions that are positive to the actual data. The F1 score value is the harmonic mean value of precision rate and recall rate, and it is a comprehensive evaluation index of precision rate and recall rate. The calculation formula is as Eq. 2.12.

$$
\begin{aligned}
\mathrm{P} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\
\mathrm{R} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \\
\mathrm{F1} &= \frac{2}{\frac{1}{\mathrm{P}}+\frac{1}{\mathrm{R}}}
\end{aligned}
\tag{2.12}
$$

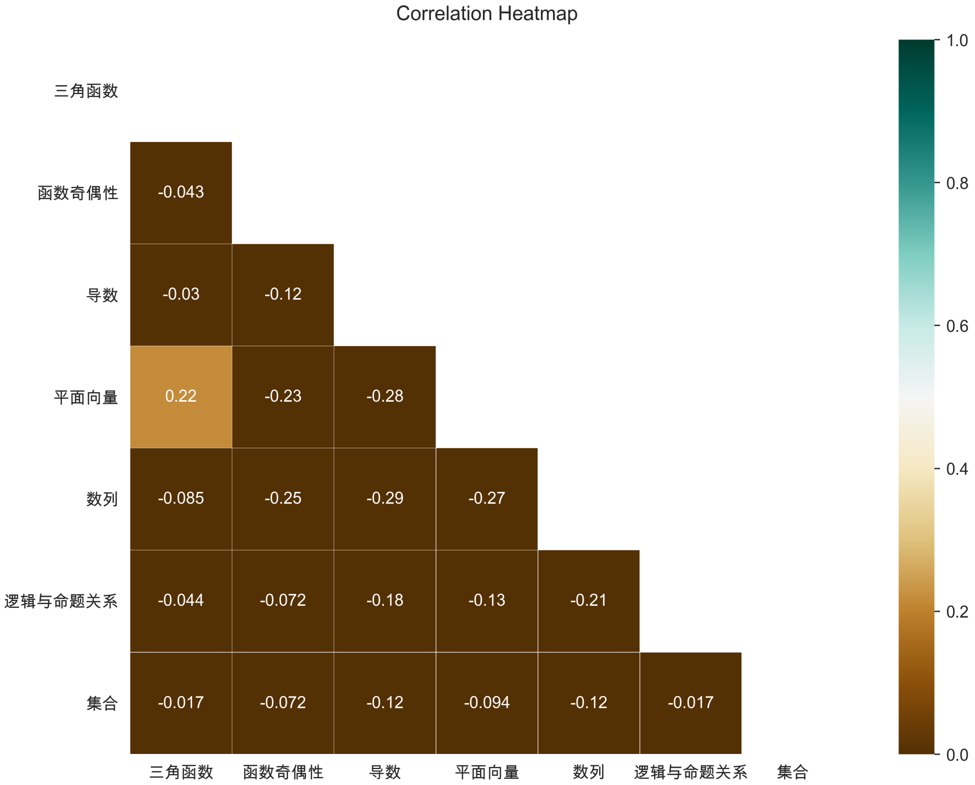


Fig. 2.15 The relation heatmap between different categories

Similarly, for the multi-label tagging problem, Macro F1 and Micro F1 can be used as the metrics. Micro-F1 first calculates the total number of TP, FN, and FP, and then calculates F1, while Macro-F1 calculates the F1 of each category separately and then averages (the weight of each category F1 is the same). The formula is Eq. 2.13 and Eq. 2.14. The c is the total number of labels.

$$\mathrm{F1}_{macro} = \frac{1}{c}\sum_{j=1}^{c}\mathrm{F1}(\mathrm{TP}_j, \mathrm{FP}_j, \mathrm{TN}_j, \mathrm{FN}_j) \tag{2.13}$$

$$\mathrm{F1}_{micro} = \mathrm{F1}(\sum_{j=1}^{c}\mathrm{TP}_j, \sum_{j=1}^{c}\mathrm{FP}_j, \sum_{j=1}^{c}\mathrm{TN}_j, \sum_{j=1}^{c}\mathrm{FN}_j) \tag{2.14}$$

Similarly, the statistic of some indicators based on samples can be obtained. For example, the multi-label accuracy rate Acc_{ML}, multi-label precision rate $\mathrm{Precision}_{\mathrm{ML}}$, multi-label recall rate $\mathrm{Recall}_{\mathrm{ML}}$ and $\mathrm{F1}_{ML}$ can be calculated. Accuracy is the proportion of samples with completely correct predicted labels in the overall sample. Hamming loss is a measure of the difference between the predicted label and the true label. Precision and Recall represent the proportion of true positive samples in true samples and the proportion of true positive samples in positive samples proportion. Their calculation formulas are Eq. 2.15-2.18.

$$\mathrm{Acc}_{ML} = \frac{1}{n}|\{i|Y_i = \tilde{Y}_i\}| \tag{2.15}$$

$$\mathrm{Precision}_{\mathrm{ML}} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap \tilde{Y}_i|}{|\tilde{Y}_i|} \tag{2.16}$$

$$\mathrm{Recall}_{\mathrm{ML}} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap \tilde{Y}_i|}{|Y_i|} \tag{2.17}$$

$$\mathrm{F1}_{ML} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{2.18}$$

2.4.4 Experiment Settings

The problem addressed by this model is the labeling of related knowledge points, so the performance of classifying exercises with related knowledge points should be evaluated in the experiment. Therefore this model is trained and will be applied on multi-knowledge point exercises for accuracy, AUC, and multi-classification metrics and compared with the baseline model.

In this experiment, the training and test sets are split from the original dataset by the ratio of 3:7, and one-tenth of the training set is taken as the validation set. All steps of this experiment are run on a stable cloud server environment to ensure the experimental environment's consistency. The running environment are shown in Table 2.3. Furthermore, there are many adjustable hyperparameters within the model, several critical ones of which are shown in Table 2.4.

Table 2.3 Experiment running environment

Software/Hardware	Configuration
CPU	Xeon Gold 6139
GPU	Tesla V100
VRAM	16G
Operating System	Ubuntu 18.04
Python	3.8.6
PyTorch	1.8.0
GPU Driver	Cuda11.2/cudnn8

Table 2.4 Hyperparameter settings of recommendation model

Hyperparameter	Value
Dimension of hidden layer	64
Number of GCN Layers	2
Max text length	128
Optimizer	Adam
Learning rate	0.001
Number of classes	7
Batch size	32
Embed size	100
Training epochs	20

2.4.5 Model Training

The training and validation loss plot are shown in Fig. 2.16 and the accuracy plots are shown in Fig. 2.17. During the training process, when better validation accuracy is encountered, the model parameters will be preserved, and when the training is finished, the saved model is the optimal model, which will be applied to the test set for model performance validation. According to the training graph, it can be concluded that the model converges after about 15 epochs of training.

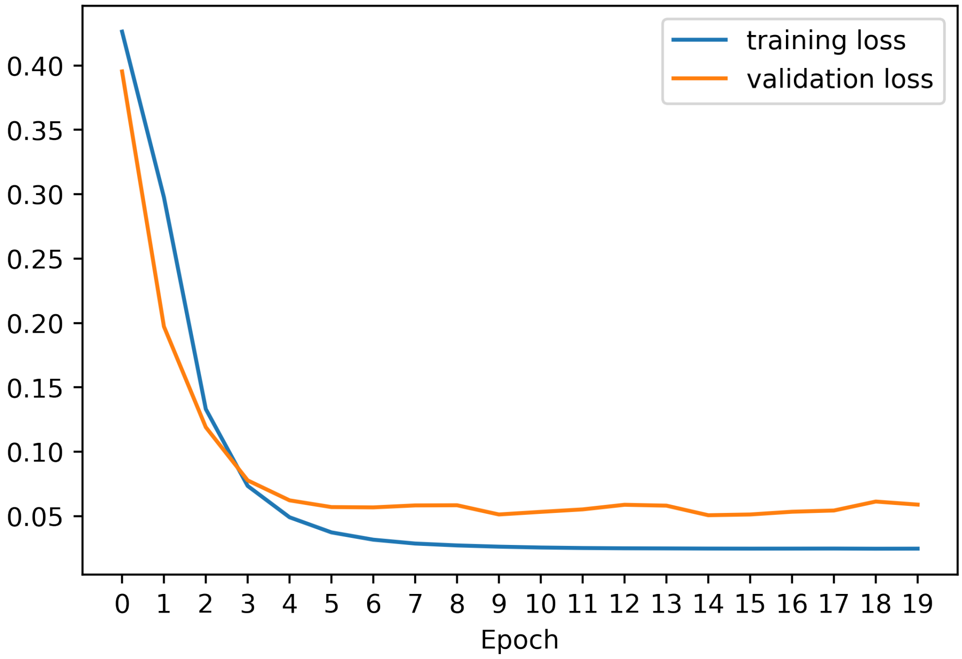


Fig. 2.16 The training and validation loss plot of model

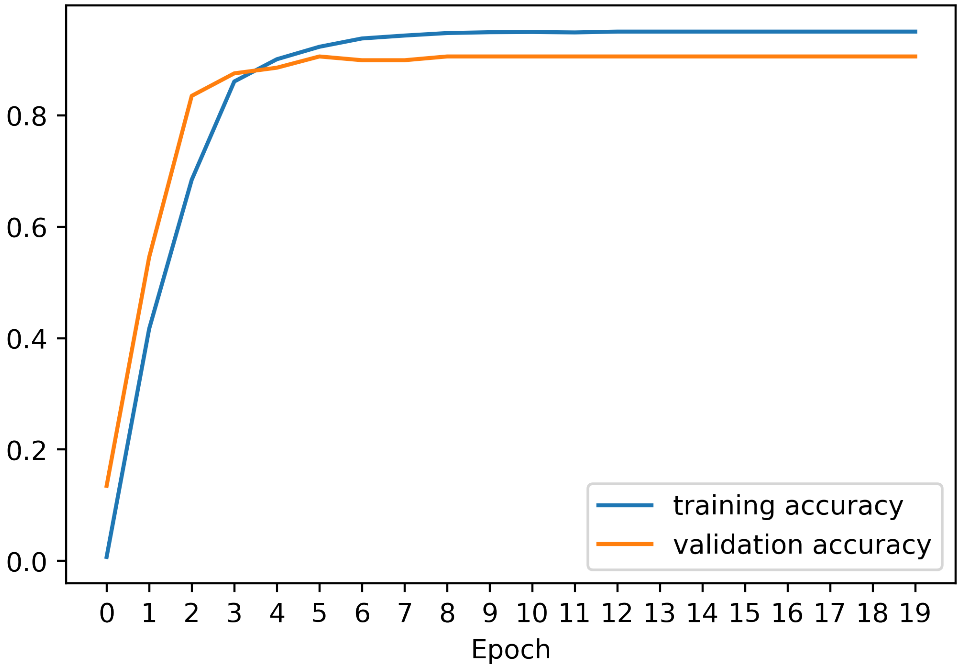


Fig. 2.17 The training and validation acc plot of model

2.4.6 Result and Analysis

In the experimental comparison for each of the other baseline models, three experiments were conducted, and the average of the experimental results was taken as the performance parameter of the baseline model and compared with the proposed model. The comparison experiment results are divided into the performance comparison of the baseline models shown in Table 2.5. The results show that the proposed model outperforms the baseline model in all the multi-label classification metrics, so the present model achieves the intended design goal. Further, the classification details of the model for each category are shown in Table 2.6.

Table 2.5 Result comparison

Metrics	$F1_{macro}$	$F1_{micro}$	Acc_{ML}	$F1_{ML}$
BiLSTM+Attention	0.824	0.924	0.874	0.926
fastText	0.846	0.922	0.854	0.916
TextCNN	0.761	0.923	0.857	0.917
Proposed	**0.912**	**0.932**	**0.888**	**0.937**

As can be seen from the Table 2.6, the performance of the models proposed in this chapter is better for exercises that occur more frequently than those that occur less frequently. As the frequency of knowledge points decreases and the difficulty of classification increases, the model shows performance degradation. This is because the model is prone to overfitting when the exercises' sample size is small, leading to the degradation of the model prediction performance. The problem can be solved by increasing the sample size and optimizing the data quality.

Table 2.6 Classification report of proposed model

Class	Precision	Recall	F1 Score	Support
三角函数	0.957	0.710	0.815	31
函数奇偶性	0.946	0.930	0.938	187
导数	0.918	0.866	0.892	247
平面向量	0.942	0.961	0.951	204
数列	0.996	0.971	0.983	243
逻辑与命题关系	0.958	0.883	0.919	180
集合	0.907	0.867	0.886	45
Micro avg	0.951	0.915	0.932	1137
Macro avg	0.946	0.884	0.912	1137
Weighted avg	0.951	0.915	0.932	1137
Samples avg	0.951	0.935	0.937	1137

2.5 Summary

In the exercise recommendation system, there are a large number of exercises with missing knowledge point labels. It is necessary to label the exercises to meet the adaptive learning system's requirements, while manual labeling is high cost and low efficiency. Therefore, automatic marking of knowledge points of test questions is a critical problem to be solved. This chapter proposes a multi-knowledge point labeling model for exercises based on GCN and Bi-LSTM text mining model based on the attention mechanism. For the proposed model, a better knowledge point labeling performance than existing models has been achieved after verifying the experimental data set.

The contributions of this chapter are as follows:

1. The relationship between knowledge points is modeled by a graph neural network, which can expose the hidden knowledge point labels in the original text and provide the existing model with the latent knowledge point reasoning capability.

2. By designing the correlation function between the knowledge points, the dependence between the knowledge points is modeled based on knowledge points' co-existence in the same exercise.
3. The proposed model achieved better performance than baseline models in multi knowledge labelling task.

The labeling of exercise knowledge points is the first step of the recommendation system. The labeled exercises can be used as the input of the knowledge tracing model to track students' knowledge state and can also be adopted as the recommendation system's input feature to recommend exercises based on knowledge points.

Chapter 3

Graph Attention Networks Embedded Knowledge Tracing Model With Transformer

3.1 Research Motivation

This section is a core part of this recommendation system to obtain the student's knowledge mastery status using knowledge tracing. The model is used to assess the student's knowledge acquisition status, which is adopted as the key factor for recommendations in the subsequent recommendation model. Knowledge tracing is a standard model used for students' cognitive diagnosis, which models students' knowledge acquisition based on their previous answer records to obtain their knowledge status [44]. There is a wealth of models for knowledge tracing, early models of knowledge tracing Bayesian Knowledge Tracing (BKT) [11], which issued the assumption that students' solutions to exercises are based on a set of knowledge points that are considered to be unrelated to each other and are therefore represented independently of each other. This approach fails to capture the relationships between different concepts nor to characterize complex conceptual transformations. In 2015, Piech et al. proposed the deep knowledge tracing model (DKT), which first applied RNN to the knowledge tracing task, which contains an implicit state of knowledge, and achieved a baseline performance over BKT at that time, and it marked the prologue of knowledge tracing research based on neural network models. However, DKT cannot output the hidden state of knowledge and is not sufficiently interpretable.

Moreover, DKT uses to store all memories in a hidden vector, and the prediction performance for long sequences is not satisfactory enough. Memory Augmented Neural Network

(MANN) [45] was proposed to address this problem, which allows the network to keep multiple hidden state vectors and read and write these vectors separately. In 2017, Dynamic Key-Value Memory Networks (DKVMN) [46] referring to the design of MANN for knowledge tracing tasks and optimizes MANN for knowledge tracing tasks with different input and output domains. DKVMN uses key-value pairs as the memory structure, which can avoid over better prediction performance relative to BKT, and DKT is achieved by using key-value pairs as the memory structure, which can avoid overfitting. The model also has better interpretability. It stores the problem-related potential concepts in the key matrix and the mastery proficiency to the concepts in the value matrix and updates the value matrix by correlating the input exercises with the key matrix.

Although the model achieved good performance, the model treated knowledge points as mutually independent points without considering the correlations. There are web-like correlations between knowledge points in the high school mathematics knowledge network, and the exercises are built based on these knowledge points. Therefore, there are also implicit and indirect correlations between the exercises and the exercises, so it is reasonable to design a model that will consider the correlations between knowledge points to characterize the model. In addition to considering the degree of mastery of the exercises' knowledge points, the students' answering behaviors can also be added to the model as additional features, adding more consideration to the model and enhancing the model's predictive performance. This thesis proposes a knowledge tracing model using graph neural networks combined with memory enhancement networks. The model transforms the knowledge point representation matrix with graph neural networks relative to the original DKVMN model and uses additional characteristics of students' answering behaviors as model inputs.

3.2 Research Contribution

In this section, an improved knowledge tracing model is proposed based on the DKVMN structure. The DKVMN model is improved based on the above two points. The DKVMN model has the following problems. First, the model does not take into account the characteristics of students, and each student's mastery of potential knowledge points will lead to different results, i.e., when two students with the same level of mastery of knowledge points may have different prediction degrees for the answer results of an exercise. Second, the model does not consider the intrinsic connection of the knowledge points, which are treated as a slot in the storage structure, and each slot is independent of others. This prevents the changes in related knowledge points caused by changes in knowledge points. In high school mathematics subjects, there are many interrelated knowledge points. Therefore, it is necessary to incorporate the domain characteristics and join the knowledge point associated learning network.

3.3 Related Theory

3.3.1 Knowledge Tracing

Knowledge tracing provides the conditions for intelligent and adaptive education, which is personalized and automated. Knowledge tracing tracks students' knowledge status from their historical learning records, predicts future learning performance and provides targeted learning coaching. The essence is to obtain the current learning status based on the student's past learning performance to predict the future learning performance. The actual practice is to analyze the data and process modeling of students' past answer records to model the current student learning status data and let the model follow the learning status of students at each stage, and predict the probability of correct answers to the exercises based on the learning status data. In subject learning, subject knowledge consists of a series of knowledge points, and students' learning status is based on their mastery proficiency to each elementary knowledge concept. The general form of knowledge tracing is that given a sequence of student answers, which consists of a series of exercises associated with a specific knowledge point, a fundamental assumption in knowledge tracing is that student performance is based on mastery proficiency to the knowledge concept. The student's performance can be applied to infer the student's mastery proficiency for each knowledge concept, i.e., the mastery proficiency to the learning state.

The mathematical representation of the knowledge tracing task is an interactive answer record $X_t = (x_1, x_2, \dots, x_{t-1})$ of a student on an exercise sequence, based on which the student's learning status is obtained by modeling and predicting the student's performance in the next exercise x_t. Where x_t is usually represented as an ordered pair (q_t, a_t), the ordered pair indicates that the student answered the exercise q_t at time t, and a_t indicates the score of the exercise, also many knowledge tracing tasks are represented by correct or incorrect answers, when a_t is 0 or 1. In this case, what is actually predicted is the probability of answering correctly for the next exercise $P(a_t = 1 | a_{t-1}, \dots, a_1)$. As shown in the figure Fig. 3.1.

3.3.2 Dynamic Key-Value Memory Networks

In traditional machine learning models, logic flow control and external memory mechanisms are often missing. As a result, many current machine learning models cannot build

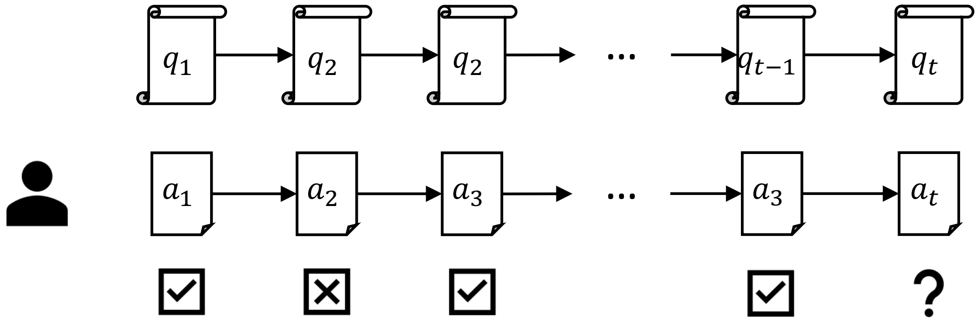


Fig. 3.1 The knowledge tracing pattern

complex logic flow control and external memory modules. Traditional models are often unable to efficiently use trainable parameters to obtain models with strong memory capabilities. Current structures like LSTM and GRU have some memory capability, but that capability is affected by trainable parameters. MANN gets rid of the connection between the number of trainable parameters and the model's memory capability by more robust representational capability. It can simulate the human brain's working memory mechanisms, such as reading comprehension, reasoning, and arithmetic processes, and capture complex structural information and establish long-range information dependencies. MANN keeps intermediate information in memory, forming an information cache structure, and uses that information for subsequent reasoning learning. A MANN network is commonly divided into three parts such as controller, storage module, and reader. The controller reads the previous stage's state and the current input to obtain the next stage's output and the controller output. The storage module is a data structure model, such as a stack, queue, or array, and at each moment, the sequence output is computed by processing the reads and the controller output through an output function.

In the knowledge tracing task, since the knowledge point representations contained in the exercises are in a different space from the knowledge point mastery proficiency state representations, it is not possible to keep the mastery degree and knowledge point representations in one storage structure, so the MANN model needs to be adapted to the knowledge tracing task. In 2017, DKVMN [46] was proposed. The DKVMN model is shown in Fig. 3.2. DKVMN saves the knowledge point representations in a matrix $M^{(k)}$ as keys by keeping the corresponding knowledge point mastery representations in another matrix $M^{(k)}$ as Values. The corresponding knowledge point mastery representations are saved in another matrix $M^{(v)}$ as values. On inputting an exercise representation q_t, the model computes the embedding representation vector k_t of q_t with the knowledge point representation module

$M^{(k)}$ to obtain the knowledge point relevance representation vector w_t of the exercise. Each exercise can be represented as a correlation vector with a knowledge point, representing the interrelationship with the knowledge point. In the DKVMN model, there is a read process and a write process.

In the reading process, the model weights and calculates the weight vector w_t with the knowledge point mastery matrix $M^{(v)}$ to obtain the user's mastery representation vector r_t for the knowledge point. And r_t is spliced with the current exercise embedding vector k_t, and f_t is derived after the full-linked activation layer loses $Tanh(\cdot)$, which indicates the user's mastery of the exercise. The f_t goes through a final $Sigmoid(\cdot)$ activation function to output a predicted probability of answering the exercise correctly p_t. During the writing process, the current actual response performance will impact the student's knowledge acquisition status, which is a re-evaluation process. The current interaction record $x_t = (q_t, a_t)$ will be used as the input to the model. Its embedding representation v_t will be used to compute the knowledge reduction vector e_t and the knowledge increase vector a_t, and the reduction and increase vectors will be weighted with w_t and act on $M^{(v)}$ thus realizing the change for the student's knowledge state. The training process is performed by minimizing the cross-entropy loss of p_t with the true label a_t of the correctness of the answers to the exercises. The whole model is microscopic and thus can be trained efficiently using stochastic gradient descent.

3.4 Proposed Model

3.4.1 Algorithm Overview

The key point of this section is the knowledge tracing of students. In this thesis, an improved knowledge tracing network model for DKVMN was proposed, which adds additional user answer features to the original model and improves the knowledge point mastery representation storage module by graph networks, thus adding inter-knowledge point relationship representation capability to the model. The model still has several sub-modules such as exercise-knowledge point association degree calculation, reading process, writing process and knowledge network propagation process, and prediction process. Its model framework is shown in Fig. 3.3.

3.4.2 Student Behavior Capturing

In this model, students' answering behavior is introduced into the model as an additional feature, forming a cross-feature with the exercise representation vector through the feature

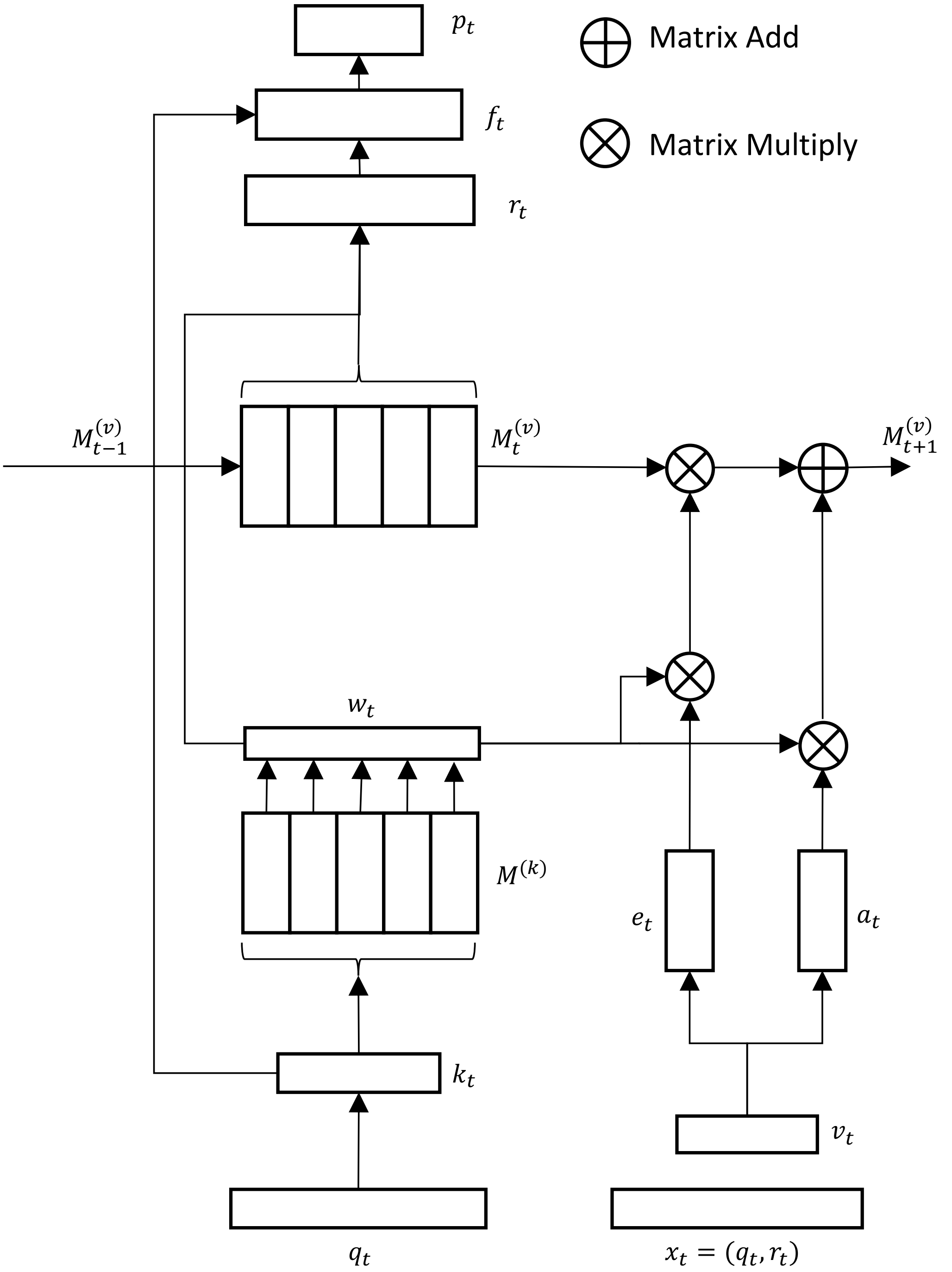


Fig. 3.2 The original dynamic key-value memory network model

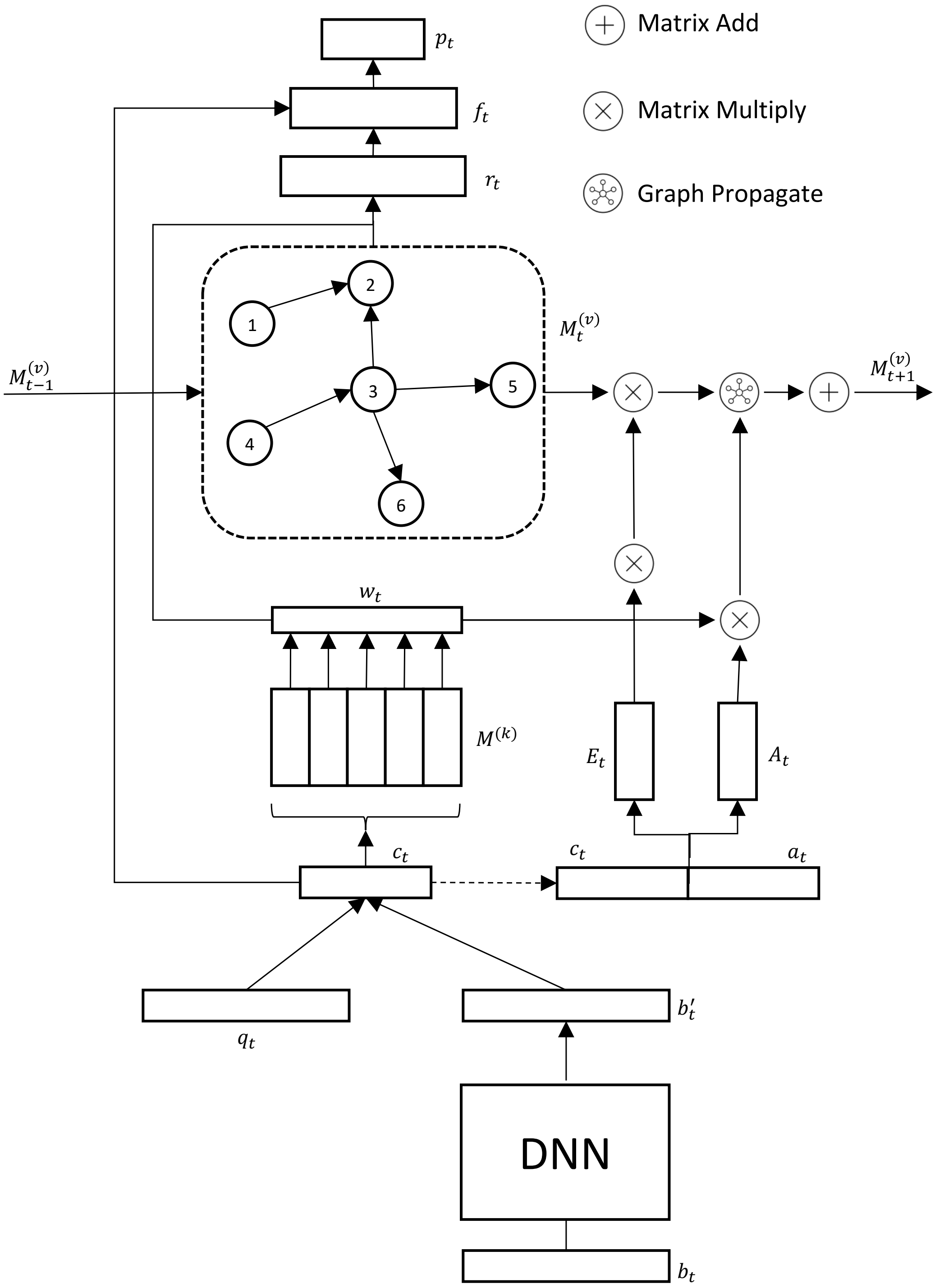


Fig. 3.3 The graph key-value memory network architecture

crossing layer. The cross-feature vector is then adapted to the subsequent calculation of the exercise-knowledge point weight vector, integrating the user's answer behavior features (e.g., hint number, overlapping time, number of attempts) vector $b_1^t, b_2^t, \ldots, b_N^t$ into a multilayer perceptron based feature extraction network and outputs a vector c_t for the user is answering ability representation. This vector is feature-crossed with the original exercise representation vector q_t to generate an exercise representation vector that incorporates user-related features. The specific process is shown in Eq. 3.1. The $||$ represents concatenation.

$$\begin{aligned} b_t' &= \mathrm{DNN}(b_1^t, b_2^t, \ldots, b_N^t) \\ c_t &= q_t || b_t' \end{aligned} \tag{3.1}$$

This mechanism introduces students' personalized answering characteristics, thus enabling the calculation of personalized weights for the exercises' knowledge points. It makes full use of neural networks' feature representing ability to extract the hidden factors that affect the correctness of students' answers, thus improving the overall reliability of the model.

3.4.3 Knowledge-Question Correlation Calculation

In the process of students' exercise practice, an exercise question is associated with multiple knowledge points, shown in Fig. 3.4. According to the cognitive diagnosis theory [47], students' mastery of the knowledge points and students' intrinsic learning ability determine the correctness of their answers. Similar to DKVMN, this model uses the Key matrix $M^{(k)}$ to keep all potential knowledge points related to the exercises. Suppose there are $N^{(K)}$ knowledge points, then $M^{(k)} \in \mathbb{R}^{N^{(K)} \times d_k}$, where d_k is the dimensionality of the knowledge point representation vector.

In Eq. 3.1, the exercise representation vector c_t incorporating students' personalized information has been obtained, where $c_t \in \mathbb{R}^{d_k}$, and the correlation $w_t(i)$ between the exercise and the knowledge point i for the ith knowledge point concept is calculated by is Eq. 3.2, where $\sigma(\cdot)$ is the softmax function.

$$w_t(i) = \sigma(c_t M^{(k)}(i)) \tag{3.2}$$

Since $M^{(k)}$ is a matrix, the correlation vector $\mathbf{w}_t = (w_t(1), w_t(2), \ldots, w_t(N^{(K)}))$ can be obtained by matrix operations like Eq. 3.3.

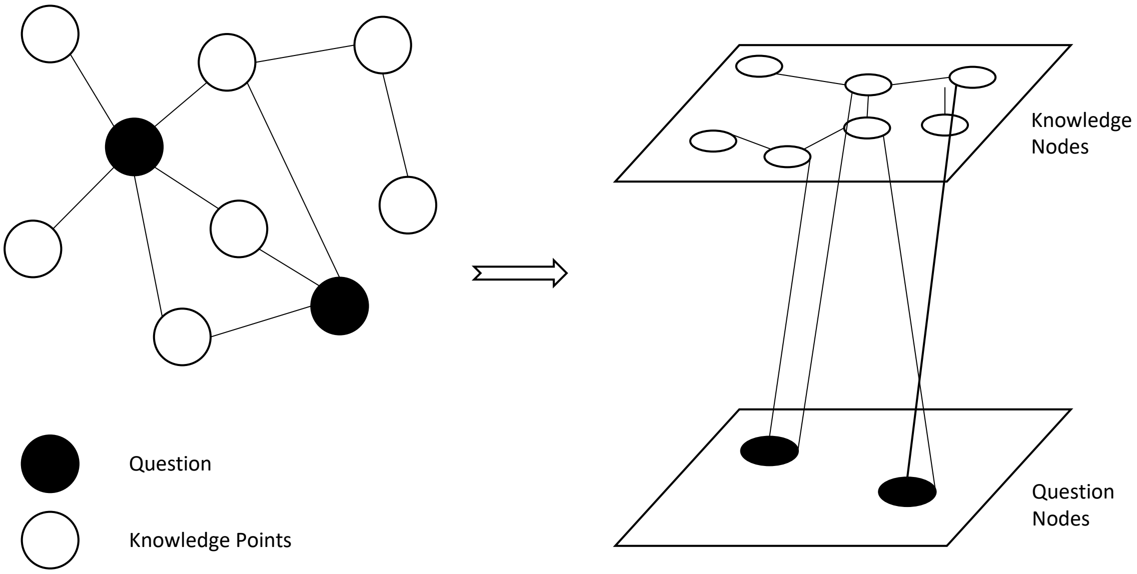


Fig. 3.4 Relation between exercise question and knowledge points

$$\mathbf{w}_t = \sigma(c_t M^{(k)}) \tag{3.3}$$

The vector $\mathbf{w}_t$ represents the relation representation of the exercises and knowledge points, and after fusing the user's answer characteristics, it can capture more influencing factors of the answer behavior.

3.4.4 Knowledge Mastery Calculation Process

The purpose of this section is to calculate the user's potential mastery of the exercise, which is performed by calculating the user's mastery of the knowledge points associated with the exercise. A matrix keeps the user's mastery of the knowledge points ($M^{(v)} \in \mathbb{R}^{N^{(K)} \times d_v}$, where d_v represents the dimensionality of the knowledge point mastery representation vector. The difference between this matrix and DKVMN is that the graph node information propagation will subsequently modify the matrix to the relevant knowledge points in terms of mastery.

The correlation representation vector $\mathbf{w}_t$ of the exercises and knowledge points calculated in the previous section is weighted and calculated with the current user knowledge mastery matrix $M^{(v)}$, as shown in equation Eq. 3.4.

$$r_t = \mathbf{w}_t M^{(v)} \tag{3.4}$$

In the model, the reading process is located before the graph propagation calculation, and its purpose is to prevent the information gain explosion caused by repeated propagation of graph information.

3.4.5 Knowledge Erase and Add Process with Graph Propagation

This section calculates the reassessment value of the user's knowledge mastery after actually answering the questions. This section incorporates a graph propagation mechanism, which takes into account the change in the proficiency of the relevant knowledge points caused by the change in the mastery of the knowledge points. The input of this section is $x_t = (c_t, a_t) \in \mathbb{R}^{(d_k+1)\times d_v}$, where a_t is the embedded representation of the correctness of students' answers to the exercise q_t, which is used together with the exercise representation vector c_t to compute the answer record embedding representation vector v_t. the knowledge acquisition increase vector A_t, and the knowledge acquisition decrease vector E_t. The formula for v_t is Eq. 3.5.

$$v_t = \mathrm{Emb}(x_t) \tag{3.5}$$

The knowledge mastery reduction vector $E_t \in \mathbb{R}^{d_v}$ is computed by v_t, which is first applied to the knowledge mastery matrix $M^{(v)}$. This calculation is done by means of a weighted sum. The weights are derived from the exercise knowledge point correlation matrix $\mathbf{w}_t$ obtained from the previous calculation, and for the ith knowledge point knowledge mastery decrease $\Delta_e M^{(v)}(i)$ and knowledge mastery increase $\Delta_a M^{(v)}(i)$ are calculated as Eq. 3.6, where $W^{(e)} \in \mathbb{R}^{d_v\times d_v}$ and $W^{(a)} \in \mathbb{R}^{d_v\times d_v}$ are the transformation matrices of E_t and A_t, respectively, and $b^{(e)}$ and $b^{(a)}$ are the corresponding biases, respectively.

$$\begin{aligned} E_t &= \mathrm{Sigmoid}(W^{(e)} v_t + b^{(e)}) \\ A_t &= \mathrm{Tanh}(W^{(a)} v_t + b^{(a)}) \\ \Delta_e M_t^{(v)}(i) &= w_t(i) E_t \\ \Delta_a M_t^{(v)}(i) &= w_t(i) A_t \end{aligned} \tag{3.6}$$

For the knowledge mastery proficiency change, each key slot of the original knowledge mastery vector is first weighted and subtracted from $\Delta_e M^{(v)}(i)$, and then the knowledge mastery gain $\Delta_a M^{(v)}(i)$ is added as shown in Eq. 3.7.

$$\begin{aligned} \tilde{M}_t^{(v)}(i) &= M^{(v)}(i)(1 - \Delta_e M^{(v)}(i)) \\ \tilde{M}_{t+1}^{(v)}(i) &= M^{\tilde{(v)}_t}(i) + \Delta_a M_t^{(v)}(i) \end{aligned} \tag{3.7}$$

Finally, changes in each knowledge point have an impact on neighboring knowledge points, so a graph network propagation mechanism is introduced here to redistribute the values across the knowledge mastery matrix. Similar to the network propagation mechanism of graph convolutional neural network, then there is the formula Eq. 3.8, where $\tilde{M}_{t+1}^{(v)}(i) \in \mathbb{R}^{N^{(K)} \times d_v}$ is the upper layer of the graph network and the input. $M_{t+1}^{(v)}(i)$ is the next layer of the graph network. $\tilde{A} = A + I_{N^{(K)}}$ is the self-connected adjacency matrix. $\tilde{D}$ is the degree matrix, i.e. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $W_t \in \mathbb{R}^{d_v \times d_v}$ is the matrix of weights to be trained. $\sigma(\cdot)$ is the activation function, and $LeakyReLU(\cdot)$ is used here.

$$M_{t+1}^{(v)}(i) \leftarrow \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{\frac{1}{2}} \tilde{M}_{t+1}^{(v)}(i) W_t) \tag{3.8}$$

It can be seen that the adjacency matrix of knowledge points is the key to graph propagation, and this thesis proposes a statistical-based approach to calculate the knowledge point adjacency matrix A, i.e., when the number of co-occurrences of knowledge points i, j exceeds a certain threshold, i, j knowledge points are set to be relevant. As shown in equation Eq. 3.9, the P_{ij} represents the probability that exercise i and exercise j exist in one exercise.

$$A_{ij} = \{ \begin{array}{ll} 0, & \text{if } P_{ij} < \tau^{(k)} \\ 1, & \text{if } P_{ij} \geq \tau^{(k)} \end{array} \tag{3.9}$$

Finally, changes in each knowledge point have an impact on neighboring knowledge points, so a graph network propagation mechanism is introduced here to redistribute the values across the knowledge mastery matrix.

3.4.6 Prediction

The computed knowledge point mastery proficiency representation r_t can be used to calculate the probability of answering the exercise correctly. It is calculated by the intermediate sum vector f_t, where f_t contains information about the mastery of the knowledge points related to the exercise and the exercise itself. The calculation formula is shown in Eq. 3.10, where the weight transfer matrix W_f and the bias vector b_f are the parameters to be trained.

$$f_t = \mathrm{Tanh}(W_f^T [r_t, c_t] + b_f) \tag{3.10}$$

Then after a fully connected activation layer, the correctness prediction p_t for the exercise q_t is output. The formula is calculated as Eq. 3.11, where the weight transfer matrix W_p and the bias vector b_p are the parameters to be trained.

$$p_t = \mathrm{Sigmoid}(W_p^T f_t + b_p) \tag{3.11}$$

The whole network is trained in an end-to-end manner j with true correctness a_t, and the overall loss function is Eq. 3.12.

$$\mathscr{L} = -\sum_t (a_t \log(p_t) + (1 - a_t) \log(1 - p_t)) \tag{3.12}$$

The knowledge mastery $M^{(v)}$ is explicit so that it can be directly used as a kind of recommendation system for judgment. Moreover, it can be subsequently used as a student knowledge mastery state analysis.

3.5 Experiments

In this section, algorithms, including the performance, are investigated through multiple dedicated knowledge tracing data sets, and the performance of the proposed baseline model is compared. This chapter first introduces and counts the data sets used, then proposes metrics, experimental settings, and operating parameter environments for model performance comparison, and finally draws the results and analyzes them.

3.5.1 Datasets

After investigating the existing knowledge tracing model, the thesis selected the more popular ASSISTment data set and Statics data set related to mathematics, preprocessed the data set, cleaned the data, obtained the training set, and performed the performance with the Baseline model Compared.

The experiment was conducted using the ASSISTment 2009 dataset, which is a collection of real educational data collected on the ASSISTment teaching platform during the 2009-2010 academic year. The dataset contains a set of Skill builder questions, where students are considered to have mastered a skill when they meet specific criteria. Once a student has mastered a skill, there will be no exercises pushed for that skill.

Data Description

In the base record of this dataset, each row is a question answering record, containing a series of features, such as problem_id, user_id, attempt_count, ms_first_response skill_id, etc. The specific description is as Table 3.1.

Table 3.1 The column heading of ASSISTment Dataset

column heading	description
problem_id	problem or exercise ID
user_id	the ID of the user
correct	a binary value, 1 represents correct while 0 represents incorrect
attempt_count	Number of student attempts on the exercise
skill_id	skill ID associated with the exercise
hint_count	Number of student attempts on this problem
first_action	The Whether or not the student asks for all hints
hint_total	Number of possible hints on this problem.

The statistical information of the dataset ASSISTment 2009 after data pre-processing is shown in the Table 3.2.

Data Preprocessing

Data preprocessing is an important antecedent step to perform model training. Some key features are missing in the raw dataset, so data cleaning should be performed first to remove these rows. For example, rows missing the key skill_id, correctness, and user_id will be excluded. Since knowledge tracing is performed on an individual level, the input sequence of knowledge tracing must belong to the same user. In the original dataset, such sequences are disordered, so it is necessary to perform aggregated sorting of the user's answer sequences based on individual users. In order to incorporate the correctness of the answers into the new

Table 3.2 Statistics of ASSISTment 2009 dataset

Statistics	Number of Students	Number of Skills	Number of Questions	Number of Records
ASSISTment 2009	4,032	124	17,748	459,092

features, feature crossover is required, where the feature crossover is performed by weighting the two features, skill_id and correct, by summing and translating them. Feature crossing can introduce nonlinearity into the model.

In this experiment, the length of the longest training sequence is set as the hyperparameter, so for the input sequence that exceeds the longest training sequence, segmentation is required.

Data Analysis

Before model training, a dependent variable analysis is performed for the dataset to filter out reasonable variables as model inputs can effectively improve model performance. In this section, several features that may impact the final are selected for visualization and analysis from the preprocessed raw data. The appropriate features are then selected for input. First, column correlation analysis can be performed on the original data source to obtain each input feature's correlation with the output result labels. The plotted column correlation heat map is shown in Fig. 3.5.

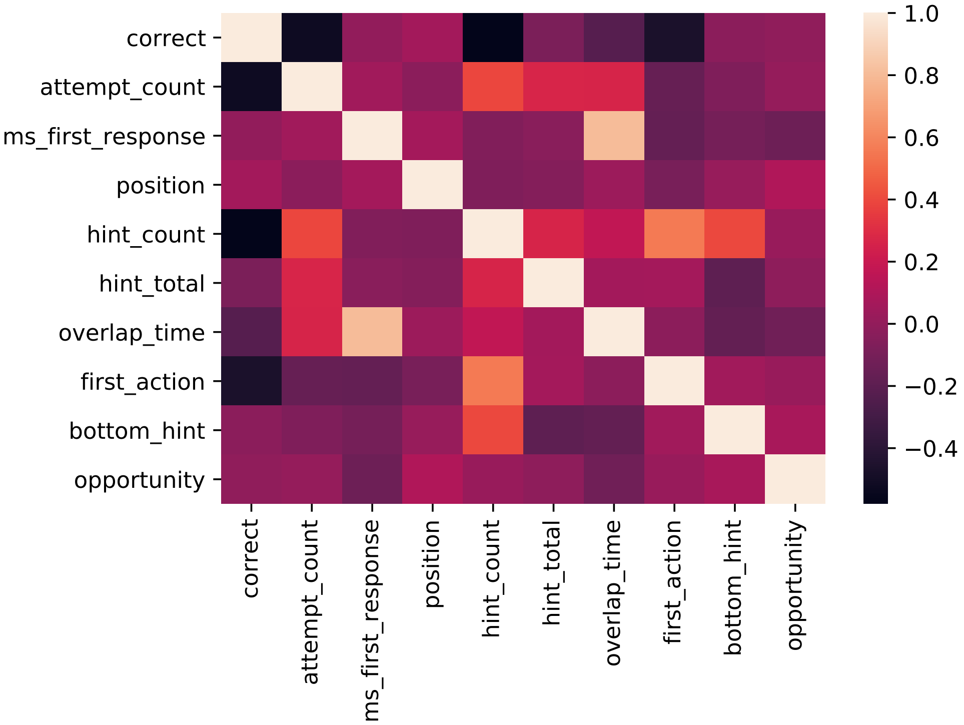


Fig. 3.5 The heat map of original dataset ASSISTment 2009

Then, combined with the heat map, the mutual correlation map and distribution map are plotted for the filtered features, respectively. The figure is Fig. 3.6. It can be seen that there is a certain mutual correlation between these features.

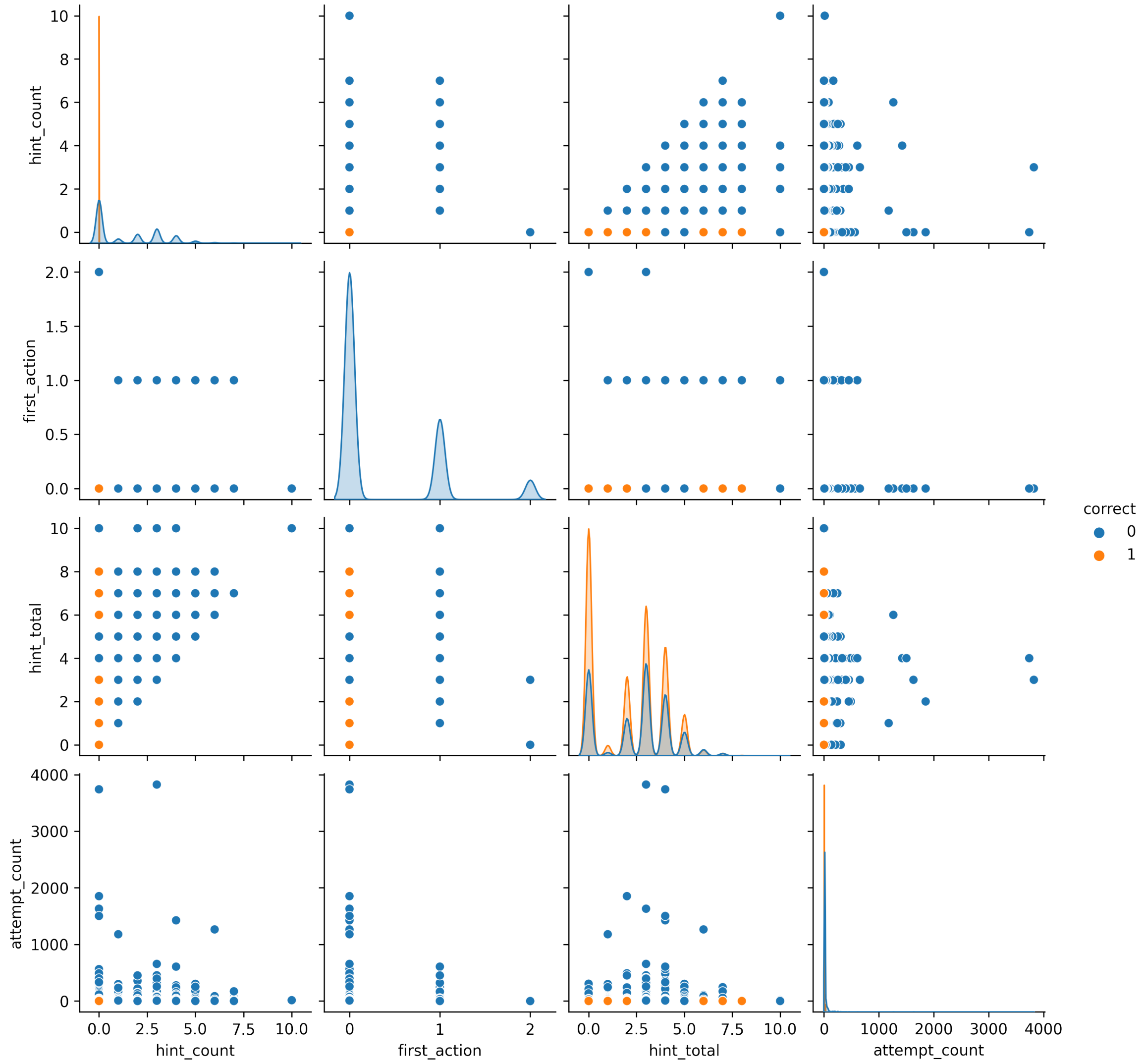


Fig. 3.6 The correlation plot between features and correctness

Next, each feature was plotted separately against the correctness of the answer, and the relationship between each feature and the target feature was analyzed, as shown in Fig. 3.7 to Fig. 3.10.

After the above analysis, the above four features can be used as additional input features for the model, which can be used as an important basis for judging the final output.

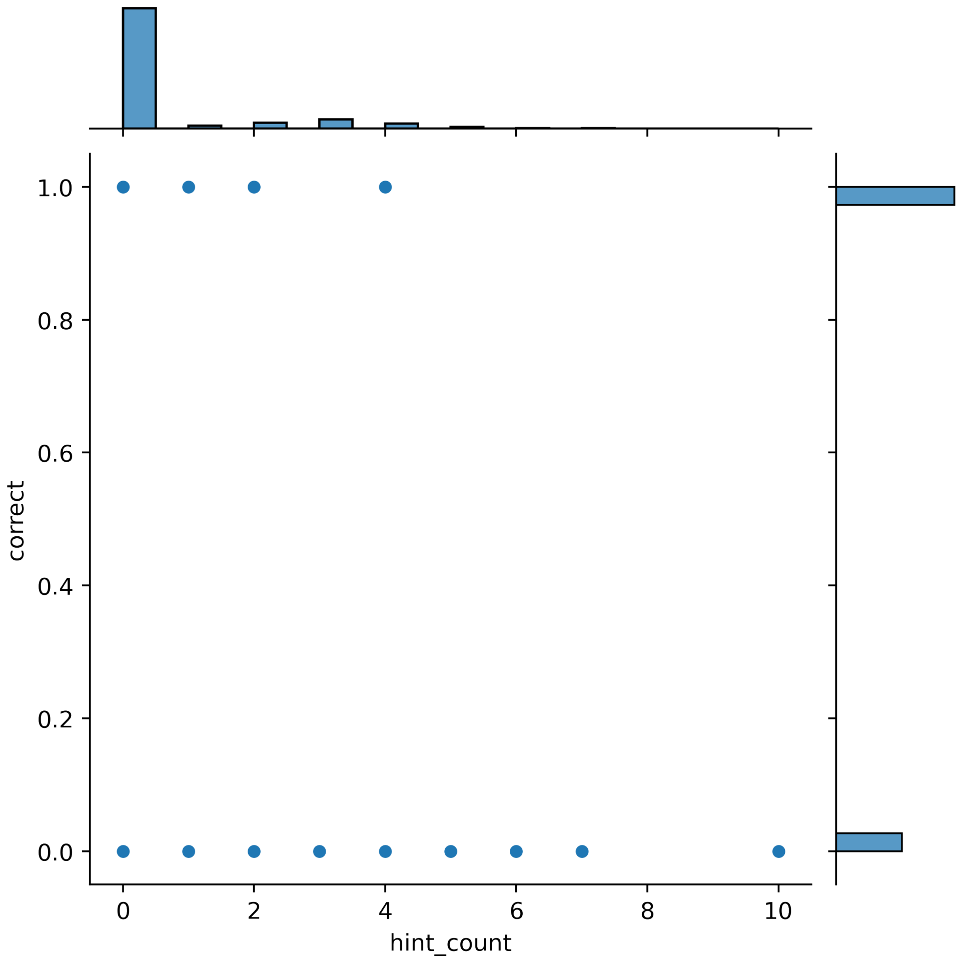


Fig. 3.7 The joint plot of hint_count

3.5.2 Metrics

Knowledge tracing is essentially a classification problem, where the commonly used classification metrics are Accuracy, Precision, Recall, F1 Score, etc. In the binary classification problem, there is the confusion matrix as shown in Fig. 3.11.

Then, the formula for these classification indicators can be represented as Eq. 3.13. The TP, TN, FP, FN represent true positive, true negative, false positive, and false negative. Although the accuracy can determine the correct overall rate, the imbalance between positive and negative sample sizes can lead to high accuracy rates that do not fully measure the system's recognition performance for both categories. Additionally, Precision represents the proportion of true positive samples predicted as positive samples, which measures the accuracy of prediction of positive sample results, and Recall represents the bi l of true positive samples identified correctly. When Precision and Recall are plotted against each other, a

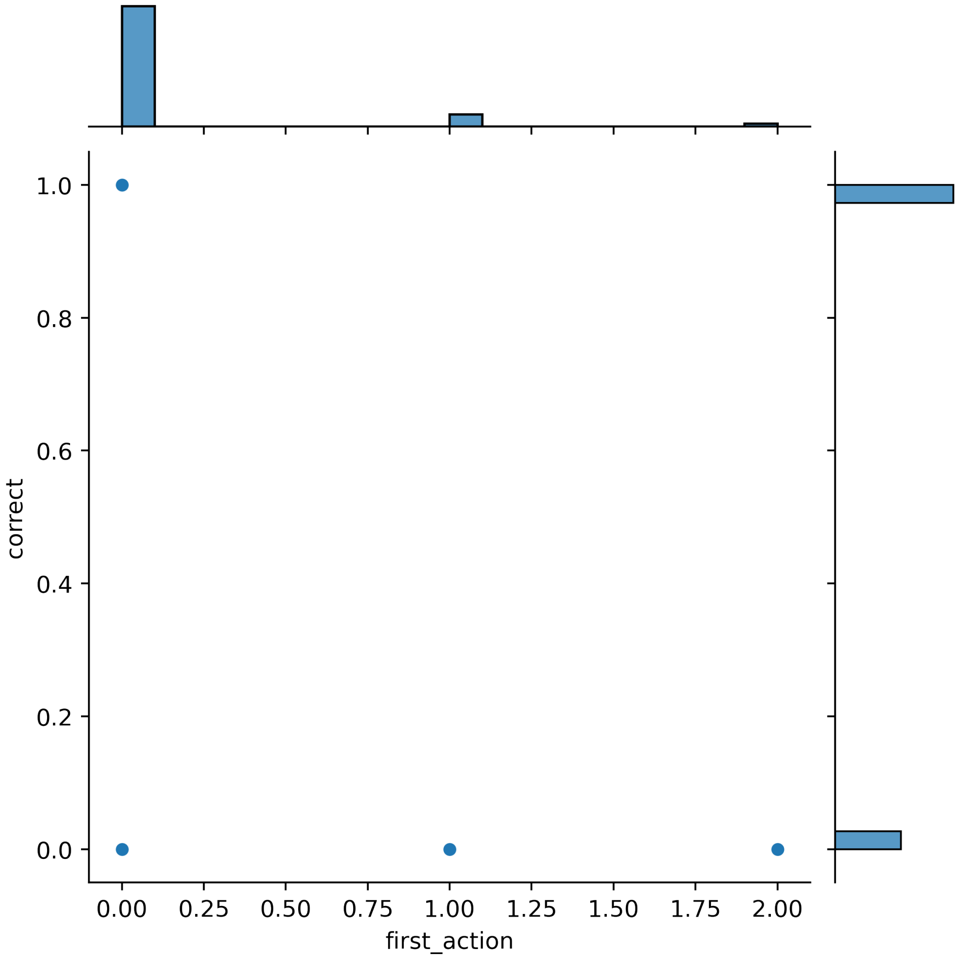


Fig. 3.8 The joint plot of first_action

P-R curve can be obtained. As Recall increases, Precision decreases overall, so a trade-off between the two is required. The F1 is a comprehensive indicator for weighing Precision and Recall.

$$\begin{aligned} \text{Accuracy} &= \frac{TP+TN}{TP+TN+FP+FN} \\ \text{Precision} &= \frac{TP}{TP+FP} \\ \text{Recall} &= \frac{TP}{TP+FN} \\ \text{F1} &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned} \tag{3.13}$$

To address the problem of sample imbalance, i.e., a large difference in the amount of correct and incorrect answers to an exercise, specific statistical characteristics need to be ap-

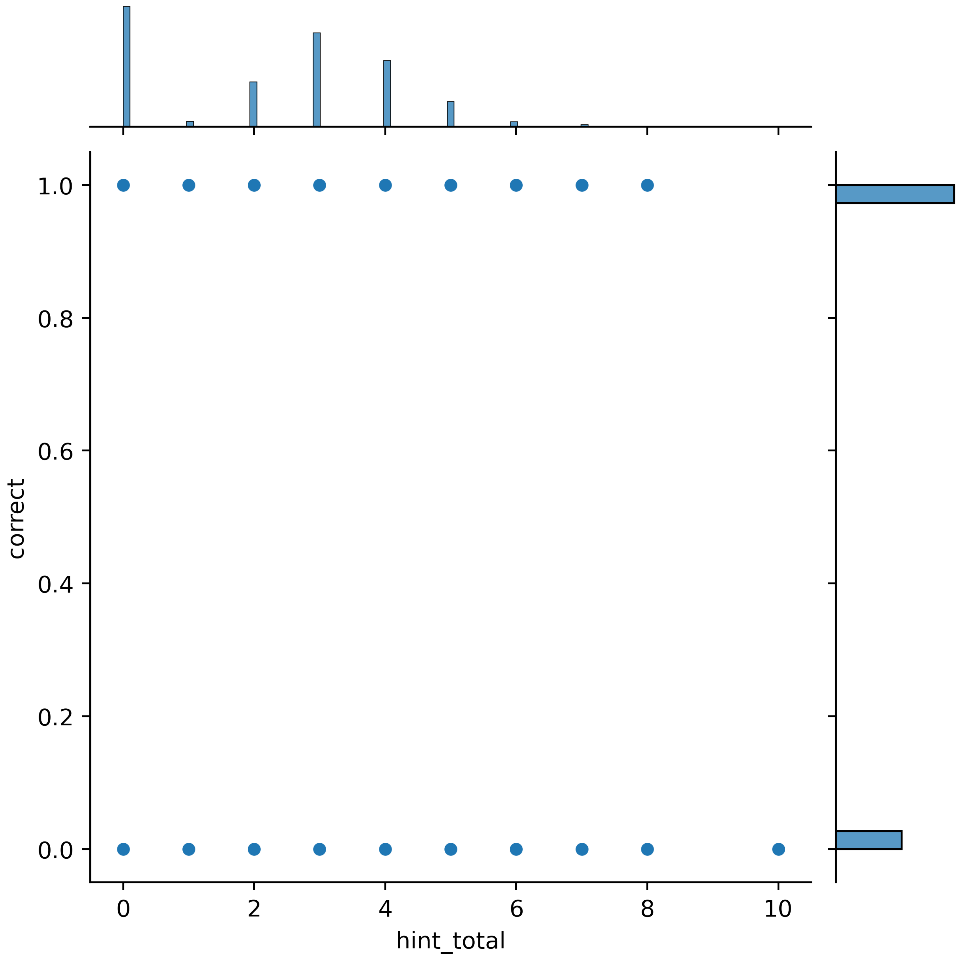


Fig. 3.9 The joint plot of hint_total

plied to the positive and negative samples, respectively. The TPR $= \mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$ and FPR $= \mathrm{FP}/\mathrm{FP}+\mathrm{TN}$ based on the actual positive and negative samples can be applied for the statistics, excluding the effect of sample imbalance. FPR indicates the degree of response that the model falsely reports, and TPR indicates the corresponding degree of coverage predicted by the model. Taking FPR as the x-axis and TPR as the y-axis, a Receiver Operating Characteristic (ROC) curve can be plotted, which evaluates the model's predictive ability. As all the classification thresholds are traversed, the FPR and TPR will change with the ROC curve. The ROC curve is also plotted by traversing all the classification thresholds, which can solve the problem of sample imbalance. Based on the above analysis, it can be inferred that when the TPR is higher and at the same time the FPR is lower, the steeper the ROC curve is, then the performance of the model is better. It can solve the problem of sample imbalance.

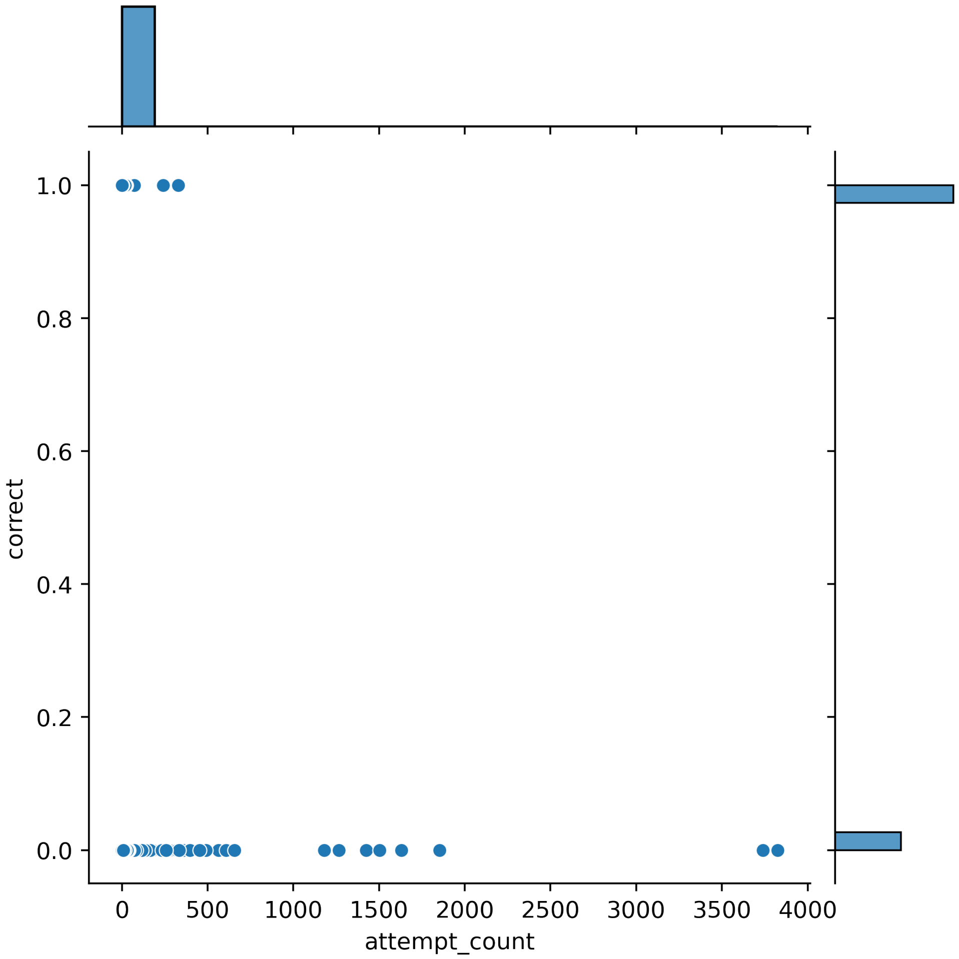
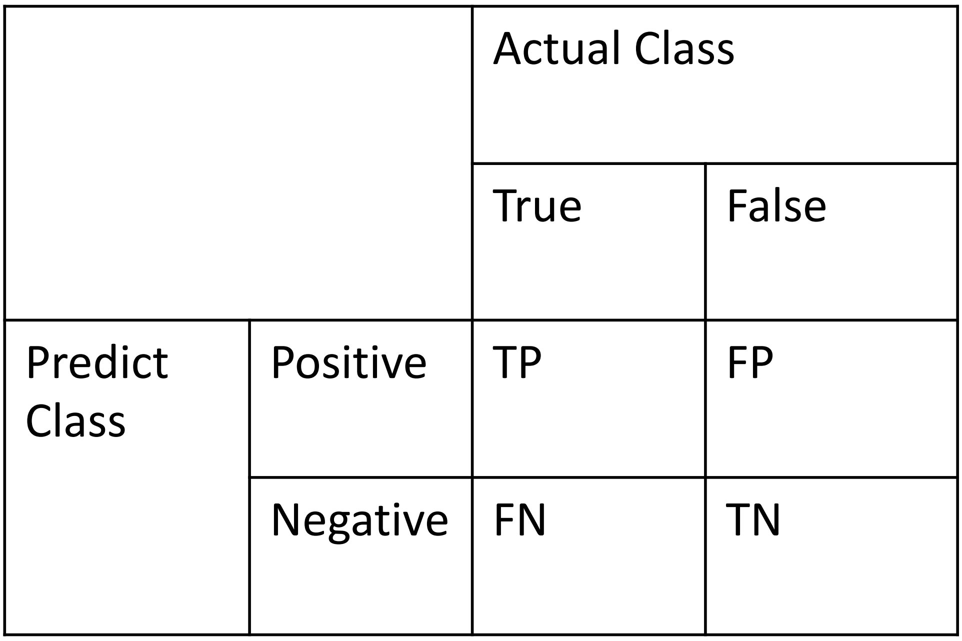


Fig. 3.10 The joint plot of attempt_count

To measure the ROC curve's steepness, Area Under Curve (AUC) can be a suitable statistic. AUC is a commonly used two-category evaluation method based on the ROC curve and has good performance comparison capabilities. The closer the AUC is to 1, the stronger the predictive performance of the model. At present, most papers in the field of knowledge tracing use AUC as a comparison indicator. In the task of knowledge tracing, the final prediction is whether the exercises at the current moment are done correctly or not, which is essentially a binary classification problem so that this approach can be adopted.

3.5.3 Experiment Settings

This experiment uses the KFold method to partition the test and validation sets, taking $K = 5$. The raw data are processed to segregate the sequence of questions done by each student, and students who exceed the predetermined longest sequence length are split into



		Actual Class	
		True	False
Predict Class	Positive	TP	FP
	Negative	FN	TN

Fig. 3.11 The confusion matrix

several sequences that do not exceed the longest length. Moreover, the experiments can use different knowledge tracing objects for cross-validation experiments, which can effectively measure the model's prediction performance.

This experiment runs on a dedicated GPU computing server, and the running environment is shown in Table 3.3.

Table 3.3 Experiment running environment

Software/Hardware	Configuration
CPU	Xeon Gold 6139
GPU	Tesla V100
VRAM	16G
Operating System	Ubuntu 18.04
Python	3.8.6
PyTorch	1.8.0
GPU Driver	Cuda11.2/cudnn8

3.5.4 Baselines

In this experiment, performance comparisons over baseline models are made by comparing some classic models with new knowledge tracing models proposed recently. The following models are evaluated in performance comparison.

- Deep knowledge tracing [12]: DKT applies the input of students' doing records into LSTM and can capture the influence of recent doing records on students' doing sequences, taking students' doing sequences into account into the model; DKT also initially considers the intrinsic correlation between knowledge points.

- Dynamic key-value memory networks (DKVMN) [46]: DKVMN uses key-value pairs as the memory structure, which can avoid over better prediction performance relative to BKT, and DKT is achieved by using key-value pairs as the memory structure.

- Self-attentive model for knowledge tracing (SAKT) [48]: The SAKT model takes into account the correlation between exercises in terms of potential knowledge points and introduces the Transformer structure to calculate the Attention between input sequences of exercises so that the relevance of current exercises can be calculated based on previous exercises and reasonable predictions can be made.

- Neural pedagogical agent (NPA) [49]: NPA applied the Attention-based Bi-LSTM model to the knowledge tracing task, which takes into account the bidirectional information of the question-answering sequence, and the subsequent answer sequence interactions also correct the previous predictions.

3.5.5 Model Training

This model requires some predefined training parameters, and after parameter adjustment, the parameter combinations shown in Table 3.4 are obtained. The experiments were conducted a total of three times, and the average training output was selected for training data plotting to obtain the loss training plot and the model performance training plot, as shown in Fig. 3.12 and Fig. 3.13. It can be seen that the model basically converges after about 150 epochs.

The results show that the model's prediction performance can converge stably after about 150 epochs of training.

Table 3.4 Hyperparameter settings of proposed model

Hyperparameter	Description	Value
lr	learning rate	0.01
opt	optimizer	Adam
batch_size	batch size	32
maxseqlen	maximum sequence length	200
q_embed_dim	exercise embedding dimension	50
qa_embed_dim	exercise answering embedding dimension	20
memory_size	number of memory slots	20

3.5.6 Result and Analysis

The experimental results are presented in Table 3.5, and the performance of the proposed model achieves optimal metrics in key indicators Accuracy and AUC compared to other models. There are two reasons for this, one is because the model considers additional features of the user and therefore introduces new influencing factors, and the other is that the model considers the model between potential concepts and uses graph-like structures to characterize such connections, thus introducing higher-order knowledge point relationship features to the model. In summary, the model introduces new explanatory features for the knowledge tracing task and therefore achieves a large improvement in the model's final performance metrics.

Table 3.5 Performance comparison between baseline and proposed models

Model	ACC (%)	AUC (%)	Training time (sec)
DKT	76.99 ± 0.08	81.79 ± 0.09	$2,731$
DKVMN	75.63 ± 0.19	79.58 ± 0.27	$3,378$
NPA	77.09 ± 0.08	81.81 ± 0.13	$3,872$
SAKT	76.37 ± 0.15	80.77 ± 0.09	$4,367$
Proposed	$\mathbf{81.34 \pm 0.25}$	$\mathbf{83.20 \pm 0.25}$	$4,597$

As for the model training time consumption, this model will have a reduced training efficiency compared to the original DKVMN model because of the additional graph information propagation process. To improving training efficiency, the graph propagation can be considered to be computed by means of matrix operations, which can make full use of the parallel computing capability of computing devices.

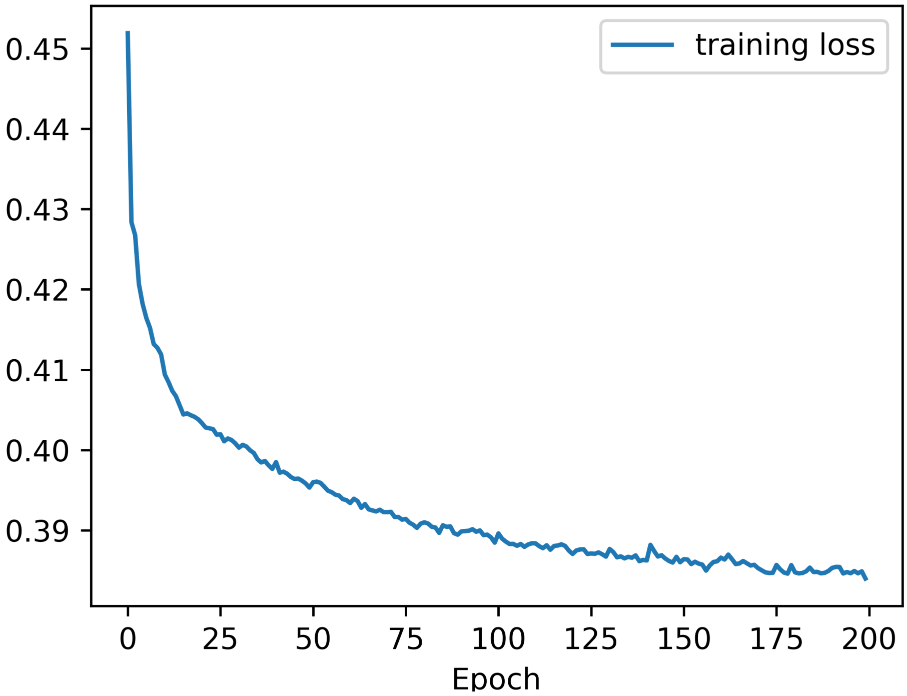


Fig. 3.12 The training process of proposed model

3.6 Summary

In this section, an improved GKVMN model based on DKVMN is proposed, which adds the ability to characterize the relationships between knowledge points to the model by transforming the storage module that integrate the exercises' potential concepts to a graph structure. On the other hand, the model introduces the student's answer characteristics to introduce more influencing factors, thus enhancing the model's predictive capability. Besides, the value storage memory slots of the model can be used as the student's knowledge point mastery representation, which can explicitly output the student's mastery of potential concepts, thus adding more recommended input features for the subsequent exercise recommendation.

The main contributions of this section are as follows.

1. the graph structure is used to characterize the relationship between the potential concepts of the exercises, taking into account the impact of the increase in mastery of the relevant knowledge points on the overall knowledge mastery. A graph propagation calculation is used to characterize the change in the mastery of the relevant knowledge points.

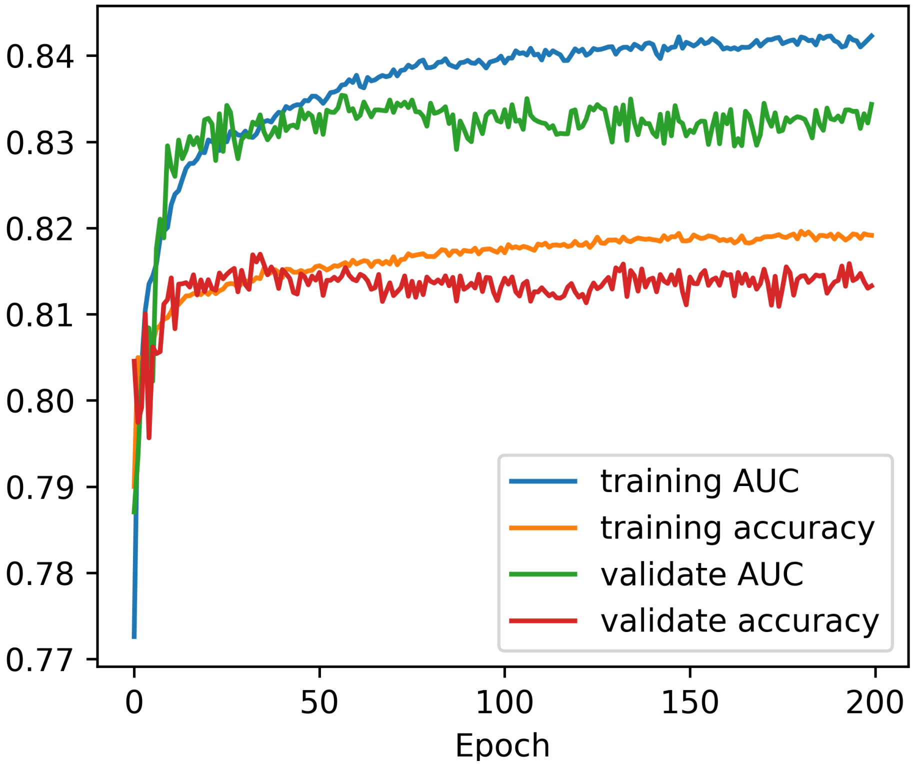


Fig. 3.13 The training acc/auc of proposed model

2. additional student characteristics are introduced as inputs to the model, which adds more predictors to the model and considers the impact of various situations on the final correctness of the answers.

3. the knowledge mastery proficiency of the model can be output explicitly, which improves the model's interpretability.

In the experimental phase, the original dataset is first cleaned and formatted with data preprocessing methods. Then the dataset is analyzed with feature correlation analysis and target-feature correlation analysis to select the best student answer features as additional input features. Finally, the performance is compared with a series of baseline knowledge tracing models. The experiments show that the proposed model outperforms the baseline model in all metrics.

Chapter 4

Knowledge Tracing Based Exercise Recommendation Model

4.1 Research Motivation

With the rapid development of today's technology, the amount of data is increasing day by day, and folks increasingly feel helpless in the face of massive amounts of data. It is precisely to solve the problem of information overload that people propose the recommendation engine technology. The recommendation system sends the recommendation algorithm to the recommendation algorithm through the user's historical behavior or the user's interest preferences or demographic characteristics. Then the recommendation system uses the recommendation algorithm to generate a list of items that the user may be interested in. At the same time, the user is passive to the search engine. At present, personalized recommendation technology is widely used in various industries. It significantly saves the cost for users to obtain information, and it also facilitates information providers to push targeted information to users. Therefore, it dramatically accelerates social information exchange efficiency, promoting media and developing industries based on information exchange, such as entertainment and education. In the education field, the application of the recommendation system is still at a relatively primitive stage.

In many cases, it still relies on manual screening of educational resources for a recommendation. This recommendation method is inefficient, poorly effective, and cannot cover all of them due to cost reasons. With the successful application of the recommendation system technology and the further maturity of the education industry, the resistance to intelligent technology in the education resource recommendation system implemented manually in the past decreases. In the context of the country's promotion of smart education, precise teaching, and smart learning, a mature and practical self-adaptive exercise recommendation system has become the education industry's expectation. This chapter proposes an exercise recommendation model based on students' knowledge mastery to improve education and teaching.

This chapter aims to recommend exercises based on the knowledge points and knowledge states of the exercises mined in the previous two chapters. The core of this chapter is to find the exercises that improve the students' current knowledge the most based on their knowledge states. The exercises are recommended to check the gaps, so it is a prerequisite for this chapter to obtain the students' knowledge status and the knowledge points involved in the exercises. However, the current exercise database is often too large, so the direct application of recommendation algorithms from the massive exercise recommendation resources will result in low efficiency, which is not conducive to large-scale commercial deployment. A portion of potentially suitable exercises is quickly filtered out from the massive exercise database as a coarse screening set based on user characteristics in the matching phase. In the sorting phase, the proposed recommendation model is applied to the coarse sieved exercise set, and a list of recommended exercises set is output in the order of priority.

4.2 Research Contribution

The main contribution of this section is to propose a knowledge-tracing-based exercise recommendation method. It applies multiple matching strategies in the matching phase to quickly generate a collection of candidate recommendation exercises, improving recommendations and reducing the recommendation system load. Besides, the control of recommendation preferences can be achieved by setting different weights of various matching strategies. In the ranking stage, the knowledge mastery of knowledge tracing is used as the basis for ranking exercises, and the exercises most probable to be mistaken can be recommended to students to understand and practice their unskilled knowledge.

4.3 Proposed Model

A recommendation system is essentially an information filtering system. Currently, common industrial recommendation systems often have several filters, ranking, and other multiple links, each filtering layer by layer and eventually filtering out dozens of items that may be of interest to users from a massive library of materials to recommend to users. As a method to solve user information overload, the recommendation system builds user portraits by analyzing user behavior data, historical records, and so on, and then recommends recommendation items that they are interested in based on the user's personalized model. In traditional recommendation systems, the models used are generally models based on matrix factorization models, such as BPR [50], factorization machines (FM) [51] and weighted matrix factorization(WMF) [52], etc. As deep learning research gradually becomes popular, deep recommendation model technology is gradually developed based on deep learning models. There are already some models that use deep learning to characterize the high-level features of recommended items, such as Deep&Wide [53], DeepCross [54], DeepFM [55]. The deep recommendation model has a partial improvement in recommendation accuracy due to its ability to model hidden features. However, in-depth recommendation algorithms often incur large computational overhead, and it is impractical to apply the full-stage in-depth recommendation model directly. Therefore, a recommendation model based on a matching-ranking two-stage can be used, and a matching algorithm with lower overhead can be used to approximate recommendation items. , Generate a set of candidate recommendation items. The matching phase's core task is to quickly obtain a pool of candidate exercises from a massive library of exercises, and the requirement is to be fast and as accurate as possible. This layer usually has a rich set of strategies and algorithms used to ensure diversity, and the algorithms need to be traded off between speed and accuracy for better recommendation results. The recommendation algorithm is then applied in the ranking stage to score or prioritize the recommended items for refined ranking. It will use the exercise, user, and exercise-user crossover features and then perform scoring and ranking by complex machine learning or deep learning models, characterized by a complex calculation but more accurate results at this layer.

4.3.1 Algorithm Overview

This chapter proposes a recommendation model based on Matching-Ranking two-stage for Math exercises. Considering that the recommended database of exercises is relatively large, the knowledge tracing recommendation model's direct application will generate a high computational cost. This section proposes a method based on multi-strategy exercises screening. The exercise screening strategy with the low computational cost is first applied to the exercises to generate the exercise recommendation candidate set, and then the recommendation algorithm is applied to the candidate recommended exercises.

The overview of the entire model is as follows:

1. Matching: The purpose of this part is to generate a series of candidate recommendation exercises. In this thesis, a multi-path matching strategy is applied, in which multiple matching strategies are applied to form corresponding sub-candidate sets and

then merged into a candidate set by the merging method. Three matching strategies based on collaborative filtering, statistical information, and user preferences are respectively applied in this thesis, and the weighted average method is used for ranking and merging.

2. Ranking: The purpose of this part is to apply the knowledge tracing recommendation model to the candidate recommended exercise set generated in the previous matching stage, sort the exercise recommendation degree according to the exercise answer's prediction, and generate a final exercise recommendation list.

The structure of the recommendation system is like Fig. 4.1

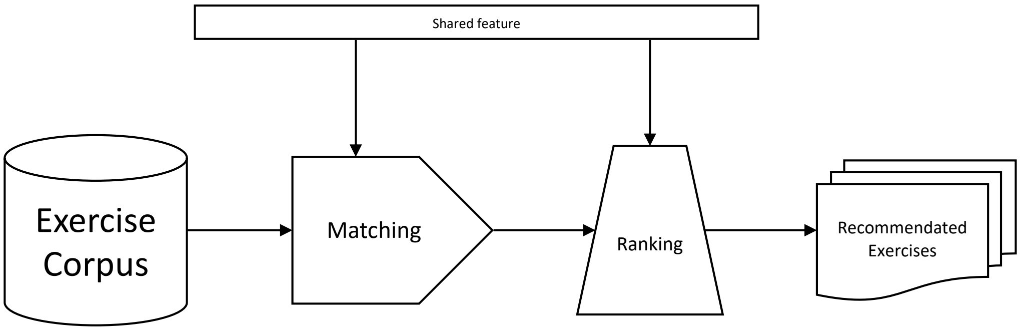


Fig. 4.1 The architecture of recommendation system

4.3.2 Matching Phase

The first stage of this recommendation model is matching. The purpose of the matching layer is to quickly filter out a set of the candidate recommended exercises while ensuring a certain degree of accuracy. There are multiple strategies for matching to deal with different application scenarios. For example, compared with news recommendation algorithms, the exercises and the user's knowledge state are more regularized vector data. It is relatively simple to calculate the similarity of exercises, users, and exercise-user cross-features. Accordingly, similarity matching can be used to apply collaborative filtering. The strategy matches recommended items. The user's historical answers to wrong exercises and statistically easy to answer exercises are of great significance for students to check their knowledge's completeness and correct their misunderstandings. Therefore, screening exercises with a higher mistaken rate can also be used as a matching strategy. Different adaptive parameters or algorithms can be applied to control the candidate problem's size to adapt to the original dataset and the recommended system deployment environment in the matching stage.

With different matching strategies, the corresponding recommendation focuses are also different. On the one hand, different matching strategies will lead to different recommendation results and ignore part of the recommendation results. For example, if the most popular exercises are emphasized, the exercises with the most individuals' records will be regarded as the matching results. If the exercises that are prone to errors are emphasized, errors will be regarded as the matching result. The exercises with a higher rate are used as the matching result. If the recommendation based on the same interest is emphasized, collaborative filtering will be used as the matching result. Adopting a certain matching strategy alone often has certain drawbacks. For example, using popularity as a matching strategy will make unpopular exercises almost impossible to be matched, resulting in an imbalance in the recommendation probability, while using the mistaken rate as a matching strategy will fail to match all students' knowledge mastery proficiency. Therefore, a single matching strategy will lose a lot of recommendation information. On the other hand, when designing the matching layer, the two indicators of "calculation speed" and "matching rate" are contradictory. The matching strategy needs to be simplified as much as possible when the calculation speed is increased. As a result, the matching rate is unsatisfactory.

Similarly, when the matching rate needs to be improved, a complex matching strategy is required. Accordingly, the calculation speed will be reduced. After weighing the two, the current industry generally adopts a ”multiplex matching strategy" in which multiple simple matching strategies are superimposed. The ”multiplex matching strategy" refers to a strategy that uses different strategies, features, or simple models to match a part of the candidate set, and then merges, filters, and other operations on these candidate sets, and then uses them for subsequent sorting models. Multiplex matching integrates multiple matching strategies and combines the advantages of each matching strategy. In multiplex matching, each strategy is irrelevant. Accordingly, it is generally possible to write multiple concurrent threads at the same time. In the actual production environment, the node cluster can be used for multi-threaded matching operations, which has the prospect of large-scale commercial applications.

This section proposes a multiplex exercise recommendation item matching model. The model uses multiple matching strategies based on popularity, user interest, expert recommendation exercises, error-prone questions, user error question sets, and collaborative filtering as sub-matching strategies. Matching strategies are based on learnable parameters to control the weight of the matching strategy. After all matching strategies have returned to the sub-matching exercise set, the exercises are merged and filtered through the weighted merging algorithm. The structure is shown in Fig. 4.2.

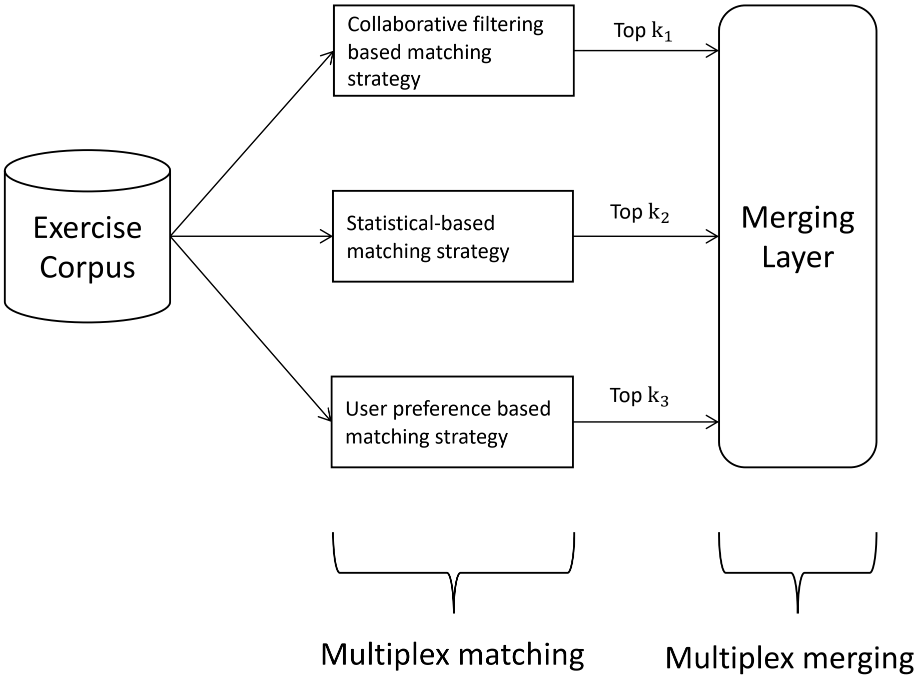

Exercise Corpus
Collaborative filtering based matching strategy
Statistical-based matching strategy
User preference based matching strategy
Top k_1
Top k_2
Top k_3
Merging Layer
Multiplex matching
Multiplex merging

Fig. 4.2 The multiplex matching method

Student-Exercise Collaborative Filtering Based Matching Strategy

Collaborative filtering is one of the matching strategies in multiplex matching. For math learning questions, because the exercises have been labeled with knowledge points, the similarity can be calculated according to the degree of overlap of the knowledge points. Simultaneously, for students, the student's knowledge state representation vector is obtained in the knowledge tracing stage, so the students' knowledge mastery similarity can also be calculated based on this vector. This section proposes a two-stage collaborative filtering algorithm based on Student-Exercise. The algorithm first calculates students' similarity, obtains related exercises, and then calculates similar exercises based on these exercises and adds them to the candidate set. The exercise set calculated by this method is relatively The language is relatively complete to prevent sparse recommendation items that appear due to fewer similar users.

Traditional user-based collaborative filtering calculates the similarity between users, determining the similarity of the user's state according to the similarity, and uses the top users with the highest similarity as the user's nearest neighbors. The construction of the neighbor user set is an important premise and foundation of the collaborative filtering algorithm. For similar users' calculation methods, most of the current personalized recommendation systems mainly use cosine similarity, modified cosine similarity, and correlation similarity.

Cosine similarity is to calculate the cosine value of the angle between the two users' rating vectors in the vector space model, and the cosine value of the angle between the rating vectors is used to express the similarity between users. The modified cosine similarity is mainly to solve the problem that users are affected by their own or other factors, and the scoring data is unstable. Compared with the cosine similarity calculation of the similarity between users, the modified cosine similarity calculation is more important because of more considerations. Be more accurate. Because the cosine similarity calculation method is too rough and does not consider the user's influence or other factors on the similarity between users, more user-based collaborative filtering recommendation systems choose to use the modified cosine similarity method to calculate the user-to-user similarity. The similarity. The formula is 4.1, where $\mathbf{S}^u = (S_1^u, S_2^u, \dots, S_n^u)$, $\mathbf{S}^v = (S_1^v, S_2^v, \dots, S_n^v)$ represent the representation vectors of the knowledge state of users u and v output by the knowledge tracing model, respectively.

$$\begin{aligned} sim(u, v) &= \frac{\mathbf{S}^u \cdot \mathbf{S}^v}{|\mathbf{S}^u| \cdot |\mathbf{S}^v} \\ &= \frac{\sum\limits_{i=1}^{n} S_i^u \cdot S_i^v}{\sqrt{\sum\limits_{i=1}^{n} S_i^{u2}} \cdot \sqrt{\sum\limits_{i=1}^{n} S_i^{v2}}} \end{aligned} \tag{4.1}$$

After the similarity of students is obtained, a similarity threshold $\tau^{simuser}$ can be set as the screening criterion for similar students. For user u, the similar user set can be denoted as $\mathbf{U}_{sim}$, the calculation of which is formula 4.2. In order to obtain similar exercises, select the user's mistaken exercise set, or favorites exercise set in $\mathbf{U}_{sim}$. Denote Aggregation of the mistaken exercise set and favorites exercise set of the user i as $\mathbf{E}_i$. Finally, the set of wrong questions and the set of favorite exercises of similar users that meet the requirements are merged and deduplicated to obtain the target exercises set.

$$\mathbf{U}_{sim} = \{i | sim(u, i) > \tau^{simuser}\} \tag{4.2}$$

The general algorithm is as Algorithm 1:

Algorithm 1 Student-exercise collaborative filtering algorithm

```
Require:
  The target student $u$;
  The similarity threshold $\tau^{simuser}$ The student knowledge state representing matrix for all $N$ students, $\mathbf{S} = \{\mathbf{S}^1, \mathbf{S}^2, \dots, \mathbf{S}^N\}$;
  The mistaken/favorite exercise sets of all students $\mathbf{E} = \{\mathbf{E}_1, \mathbf{E}_2, \dots, \mathbf{E}_N\}$
Ensure:
  The output exercise set $\mathbf{E}^{(UCF)}$
  Calculate the similar user set $\mathbf{U}_{sim} = \{i|sim(u,i) > \tau^{simuser}\}$
  Aggregate the mistaken/favorites exercise set of users in $\mathbf{U}_{sim}$, $\mathbf{E}^{(UCF)} = \sum_{i \in \mathbf{U}_{sim}} \mathbf{E}_i$
  return $\mathbf{E}^{(UCF)}$;
```

The exercise E_j of in students exercise set $\mathbf{E}_i$ recommendation score denoted as $S^{(UCF)}$ is defined by formula 4.3. The strategy weight is denoted as $w^{(UCF)}$.

$$S_{i,j}^{(UCF)} = sim(u,i) \tag{4.3}$$

Popularity-based Matching Strategy

This section proposes a recommendation item matching strategy based on the statistical popularity of the exercise corpus. The strategy focus on the exercises' popularity consists of click rate, favorite rate, mistaken rate, and expert recommendation. Click rate is a feature that combines the number of times the exercises have been answered. The favorite rate is the number of times users favorite the exercises. Mistaken rate is the number of mistakenly answering to the exercise. The comprehensive popularity score is denoted as $S^{(PMS)}$, of which the number of answering record to the exercises and the corresponding weight is denoted as $N^{(SA)}$ and $w^{(SA)}$, the number of favorites to the exercise and the corresponding weight are respectively denoted as $N^{(SF)}$ and $w^{(SF)}$, the times of incorrect answers record and the corresponding weight are denoted as $N^{(SW)}$ and $w^{(SW)}$. The weight value is adjusted according to the focus of the source. Furthermore, expert-recommended exercises should be concerned, which are some excellent exercises in the set of exercises manually recommended by industry practitioners with educational experience based on experience, with a recommendation score $S^{(EX)}$ and respective weight $w^{(EX)}$. Then the comprehensive popularity calculation formula of one exercise is 4.4.

$$S^{(PMS)} = w^{(SA)} \cdot \sigma(N^{(SA)}) + w^{(SF)} \cdot \sigma(N^{(SF)}) + w^{(SW)} \cdot \sigma(N^{(SW)}) + w^{(EX)} \cdot S^{(EX)} \tag{4.4}$$

Where $\sigma(\cdot)$ represents the sigmoid function.

After calculating $S^{(PMS)}$, sort the exercises according to the size of $S^{(PMS)}$ from large to small, and filter the first $K^{(PMS)}$ items as the recommended item matching set. Here $K^{(PMS)}$ is an adjustable hyperparameter, which can be adjusted according to the needs of the business to control the recommended focus and can be adjusted according to the needs of the business to control the recommended focus.

User-Preference based Matching Strategy

This section proposes a matching strategy for recommended items based on user preferences. This strategy takes the error answered or favorited exercises of the user as a matching factor. In this way, users can customize the recommended results of exercises and choose exercises based on their own assessment of their knowledge state. Concerning one exercise, the number of wrong answering is denoted as $N^{(UW)}$, the knowledge mastery degree is calculated as $M^{(UW)} = \frac{1}{N^{(UW)}}$. The recommendation score $S^{(UW)}$ can be calculated as $S^{(UW)} = \sigma(M^{(UW)})$ with correspond weight $w^{(UW)}$. In the exercise recommendation system, to enhance the user's memory and mastery of knowledge concepts, the priority of exercises with more error answering logs should be higher. The favorite score $S^{(UF)}$ is a binary value, i.e., $S^{(UF)} \in \{0, 1\}$ whose weight is denoted here as $w^{(UF)}$. The comprehensive priority of the exercises $S^{(UPM)}$ is calculated as formula 4.5.

$$S^{(UPM)} = w^{(UW)}\sigma(\frac{1}{N^{(UW)}}) + w^{(UF)}S^{(UF)} \tag{4.5}$$

Where $\sigma(\cdot)$ represents the sigmoid function.

After calculating the value of $S^{(UPM)}$, the error answered or favorited exercises can be matched. Sort the exercise by each one's $S^{(UPM)}$ and filter out the top K_{UMS} items, of which K_{UMS} is adjustable hyperparameters. If the set of user exercises is small, K_{UMS} can be set to a number larger than the set size so that all user-preferred exercises are added to the filtered set of exercises by default.

Mulplex Merging Method

Different recommendation candidate item sets are generated by applying different matching strategies, and each set has a different focus. In the merging stage, each matching set needs to be further merged and filtered. For matching strategy s_i, the first K_{s_i} candidate items will be filtered out. The K_{s_i} of each strategy is a hyperparameter, the value of which should be set based on practical evaluation and adapted to actual application scenarios to achieve the best matching effect.

The final matching set of recommended items is generated by the merging and sorting of various matching strategies. Each strategy's matching results can vary within different strategies for a specific item, so the matching score should aggregate all sub-matching strategies' recommendation scores. Denoting the recommending score of an item i in the strategy j as $S_{j,i}$ and the merging weight of the strategy j as w_j, the final merged matching ranking score of the item is calculated as formula 4.6. Finally, the exercises are sorted in decreasing order according to the recommended score S_i, and the top K_{Merge} items are taken.

$$\begin{aligned} S_i &= \sum_j w_j \cdot S_{j,i} \\ &= w^{(UCF)} \cdot S_i^{(UCF)} + w^{(UPM)} \cdot S_i^{(UPM)} + w^{(PMS)} \cdot S_i^{(PMS)} \end{aligned} \tag{4.6}$$

After multi-channel matching merging settlement, a candidate exercises set $\mathbf{E}^{(Candidate)}$ of size K_{Merge} is generated, each of which contains a corresponding weight and source to calculate the recommended loss function. Since the purpose of matching is to provide candidate recommendations for the final ranking stage, a final recommendation weight will be output for each exercise in the ranking stage to construct the final recommendation list.

4.3.3 Ranking Stage

After the exercise set's recommended matching stage, a candidate set of exercises is obtained, which contains exercises selected from various matching strategies. Based on the knowledge tracing model proposed in the previous chapter, this section proposes a method for recommending and sorting exercises based on knowledge state prediction.

The knowledge tracing model outputs a prediction of the correct probability of answering the exercises. Since the purpose of exercise recommendation is to strengthen students' proficiency for unfamiliar and relatively low-level knowledge points, the recommended exercises should be made as far as possible to allow students to answer incorrectly. Otherwise, it will not be able to have the effect of checking for missing knowledge points. In the knowledge tracing model, at the interaction time t, the student's knowledge state representation vector h_t can be obtained. In order to rank the degree of the recommendation of exercises, all exercises in the recommended candidate set of exercises $\mathbf{E}^{(Candidate)}$ should be input into the knowledge tracing model to predict the answers to the exercises. Make the recommended list of some of the most error-prone exercises. Its structure is shown in the Fig. 4.3.

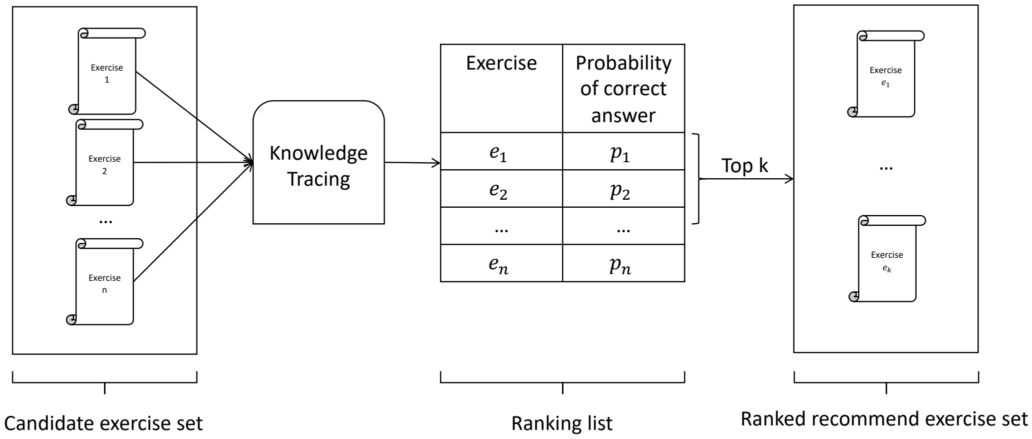


Fig. 4.3 The ranking method

The last step of the model is to input the candidate's exercises into the knowledge tracing model to get the predicted value, that is, the probability of answering the exercises correctly. Sort them in ascending order, and filter out the top K_{Rank} items as a list of recommended exercises. The K_{Rank} is an adjustable hyperparameter, which has a more significant impact on the recommendation result and determines whether an exercise will appear in the final recommendation list. When K_{Rank} is larger, the recommended accuracy rate will be higher, but the recommendation efficiency will be reduced. In practical applications, students' knowledge mastery level and practical application scenarios should be considered comprehensively, and a reasonable value of K_{Rank} should be set.

The general algorithm is as Algorithm 2:

Algorithm 2 Recommendation ranking algorithm

```
Require:
  The candidate exercises set $\mathbf{E}^{(Candidate)} = \{E_1, E_2, \dots, E_N\}$;
Ensure:
  The ranked recommendation exercise list $\mathbf{E}^{(RANK)}$
  Input each exercises $e_i \in \mathbf{E}^{(RANK)}$ to recommendation ranking model, i.e., knowledge tracing model, output the correct answering rate $p_i$
  sort $e_i \in \mathbf{E}^{(RANK)}$ by $p_i$, output a ranked exercise list$L$.
  return  $L[1..K_{RANK}]$ as $\mathbf{E}^{(UCF)}$;
```

4.4 Experiments

In the exercise recommendation system, the design principle is based on recommending as many error-prone exercises as possible given the student's state of knowledge. Therefore, in the experimental phase, a publicly available dataset for knowledge tracing is applied. The model's recommended performance is determined by estimating the model's prediction probability for the current mistaken question in a way that converts the experiment into a dichotomous task. Since the matching phase is a rule-based model, it does not have an impact on the prediction performance of the subsequent models. In the experimental phase, the performance of the knowledge-tracking exercise recommendation model in the ranking phase is mainly evaluated.

4.4.1 Dataset

Considering that this recommendation system needs to recommend exercises based on students' knowledge status, while there is no existing dedicated dataset for mathematics exercise recommendations based on knowledge mastery proficiency. Thus, in this experiment, recommendation data sets were generated from the public knowledge tracing dataset STATICS, consisting of a series of questions containing several knowledge point labels attached with related students' answer records.

4.4.2 Metrics

The goal of the recommendation model proposed in this section is to present the user with exercises that best match the current state of knowledge, so the recommendation's accuracy is a crucial metric. To intuitively show the knowledge enhancement effect of the recommendation algorithm on users, this experiment takes the interaction sequence of the knowledge tracking dataset as the input of the knowledge tracking model. It outputs the probability p_t of answering correctly for these exercises. The actual answering correctness is a_t. When an exercise is predicted to be answered correctly by the model, the exercise will not be recommended; otherwise, the exercise should be recommended. This is a classical binary classification problem. The confusion matrix is shown below.

- True Positive (TP): $p_t < 0.5$ and $a_t = 0$
- False Positive (FP): $p_t < 0.5$ and $a = 1$
- False Negative (FN): $p_t \geq 0.5$ and $a_t = 0$

- True Negative (TN): $p_t \geq 0.5$ and $a_t = 1$

As it is a classification problem, the Receiver Operator Characteristic Curve (ROC) curve can be used to determine the recommendation system's performance. Then, (Area Under Curve) AUC and Accuracy (ACC) metrics can be used to determine the model's performance.

4.4.3 Experiment Setting

For a sequence of answers, the knowledge tracing model proposed in the previous chapter outputs the probability of correct answers for each exercise, and then the model can be treated as a dichotomous classification. Therefore, a knowledge tracing model is experimentally trained and used as a predictor when the model converges. The sequence of student's answer records is input and the probability of a student's correct answer at the moment t is output.

To ensure the environment's consistency, this thesis runs on a cloud server with the running environment shown in the Table 4.1. In this experiment's configuration, 75% of the interaction records are selected as the training set and the remaining 25% as the test dataset. Run the model 5 times and take the average value. The hyperparameter settings of the recommendation model are shown in Table 4.2.

Table 4.1 Experiment running environment

Software/Hardware	Configuration
CPU	Xeon Gold 6139
GPU	Tesla V100
VRAM	16G
Operating System	Ubuntu 18.04
Python	3.8.6
PyTorch	1.8.0
GPU Driver	Cuda11.2/cudnn8

4.4.4 Model Training

The model is deployed to the cloud server and run, with a training time of about 20 minutes. The loss and accuracy graphs of the training are shown in Fig. 4.4 and 4.5. During the training process, the model saves the optimal model parameters according to the loss of validation to prevent overfitting. As can be seen from the figure, the model basically

Table 4.2 Hyperparameter settings of recommendation matching model

Hyperparameter	Value
batch size	32
epochs	200
dropout rate	0.2
window size	100
sequence len	200
question embedding dimension	50
memory size	20
QA embedding dimension	200

converges after about 20 rounds of training, and the training set produces overfitting phenomenon. Next, as the optimal parameters of the model are preserved, the validation and test losses are stabilized at the optimal level.

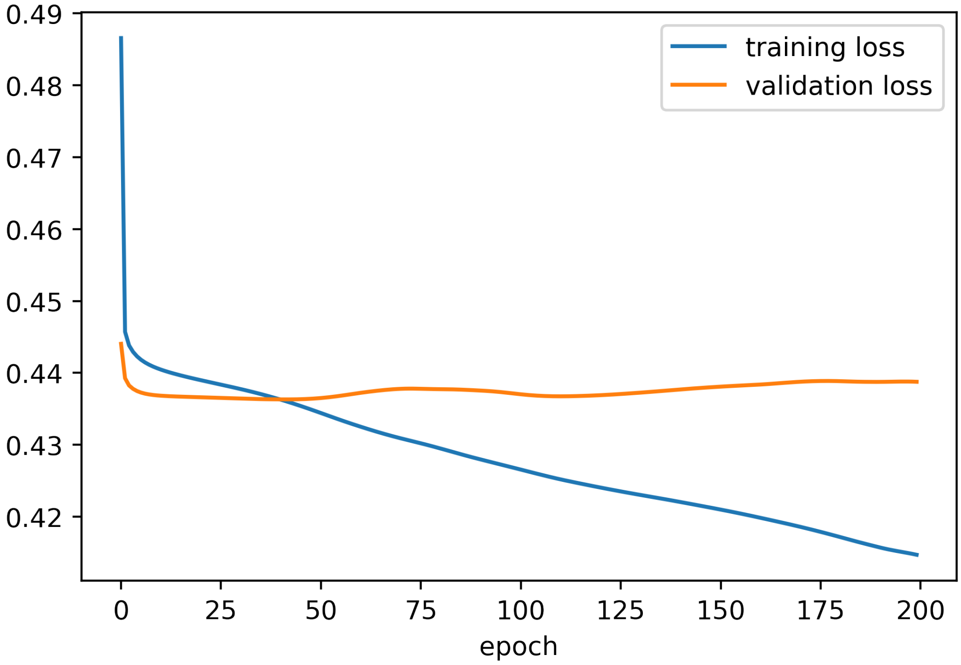


Fig. 4.4 The training loss plot

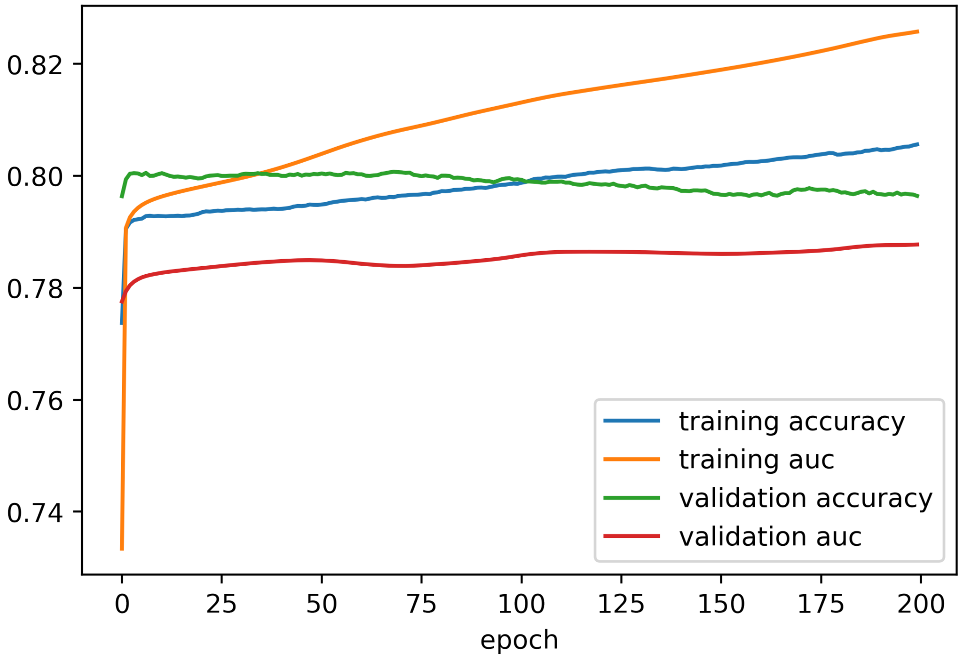


Fig. 4.5 The training accuracy plot

4.4.5 Baseline

As a comparison, the more popular Collaborative Filtering (CF) recommendation algorithm and some other knowledge tracing models are chosen as the baseline model for overall comparison.

- Collaborative filtering (CF): As a technique widely used in recommendation systems, this technique recommends the same recommendation items to similar users by analyzing the similarity between users, and in this system, the similarity is calculated by using the overall correct answer rate of students.

- Deep knowledge tracing (DKT): The original DKT model is applied to the recommender system as a knowledge tracking module, which is a model that uses an RNN structure for the knowledge tracking task.

4.4.6 Result and Analysis

The performance comparison result is shown in Table 4.3. Also, the Precision-Recall plot and ROC curve of proposed model is shown in Fig. 4.6 and Fig. 4.7.

The results show that the proposed model exceeds the baseline model in the accuracy indexes of recommendations, i.e., AUC and Accuracy, and the absolute recommendation

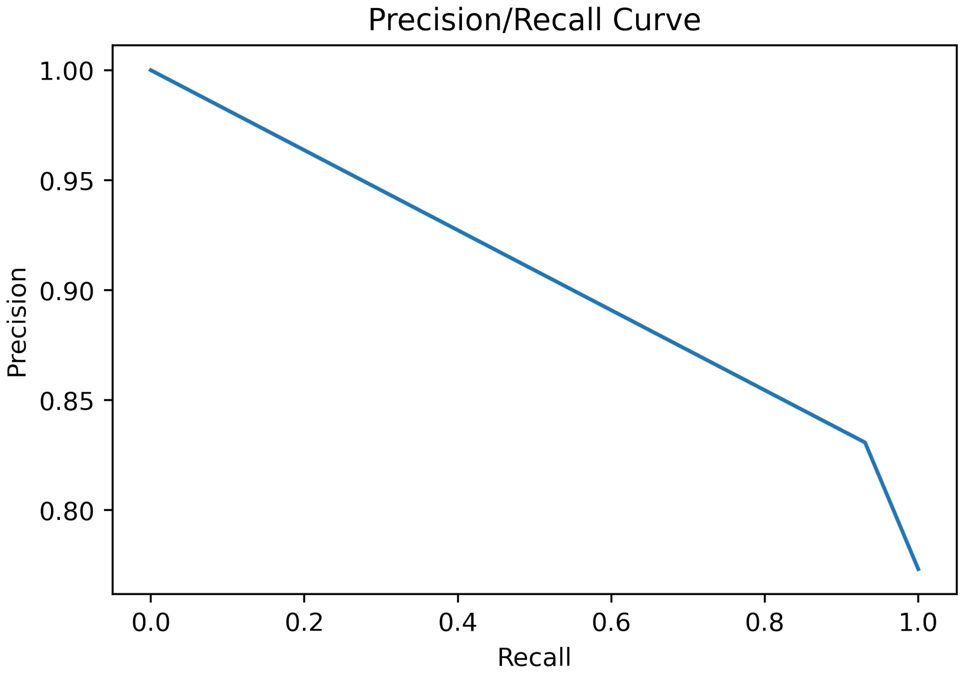


Fig. 4.6 Precision-Recall Curve of proposed recommendation matching model

Table 4.3 Recommendation experiment result

Model	ACC	AUC
CF	0.6329	0.6627
DKT	0.7741	0.7906
Proposed	**0.7997**	**0.7923**

accuracy is also higher. It can be seen that the proposed model has good classification performance, i.e., the ability to recommend error-prone questions from the exercises, indicating that the proposed model can effectively filter out the set of exercises with greater gain for students' knowledge state, thus realizing adaptive exercise recommendation based on students' learning state.

4.5 Summary

In this chapter, a two-stage Matching-Ranking based mathematical exercise recommendation model is proposed. The Matching model in the first stage is a multiplexed exercise recommendation item matching model, which employs multiple sub-matching strategies based on collaborative filtering, popularity, user preferences for generating exercise recommenda-

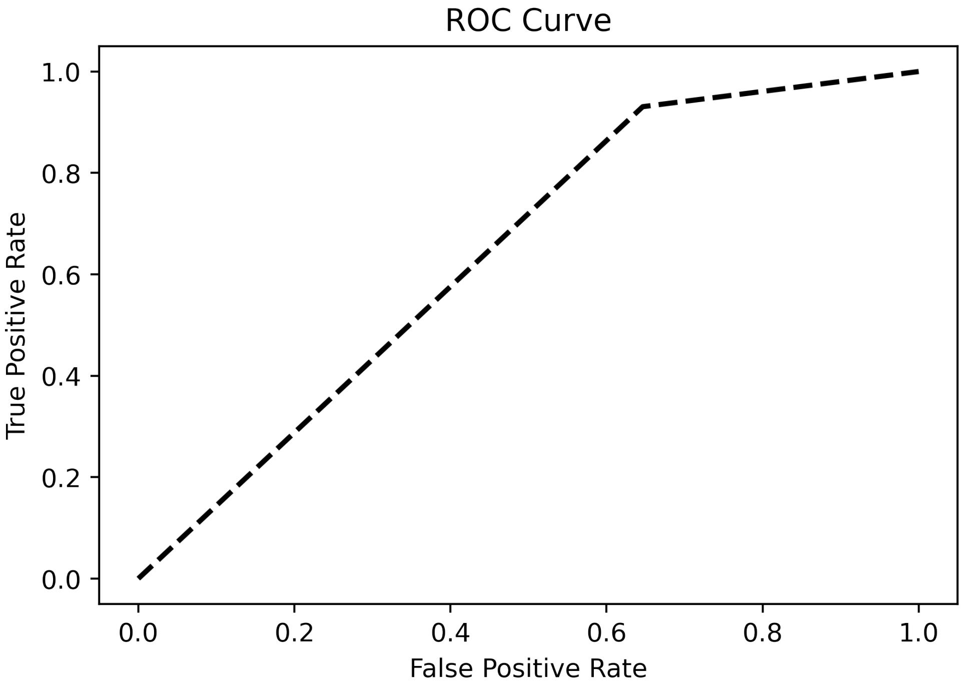


Fig. 4.7 ROC Curve of proposed recommendation matching model

tion candidate sets separately, and then these candidate sets are weighted and fused in the fusion stage to form a final fused exercise recommendation candidate set. In the ranking phase, each exercise in the set of exercise candidates obtained in the Matching phase is input to the knowledge tracing exercise for correctness prediction, and the most error-prone exercises are used as the recommendation item with the highest priority, and the recommended set of exercises is generated according to this strategy. The main contribution of this chapter is:

1. The multiple matching strategies are applied to the Matching stage of the exercises so that the candidate set of exercises can be filtered quickly with a relatively small computational load and all the exercises have different recommendation scores in different strategies. For different strategies, their weights can be dynamically adjusted to achieve manual control over the recommendation focus.

2. The knowledge tracing model proposed in the previous chapter is outputted for the exercises, and the correctness of the model output is used as the basis for the recommended score for the exercises. This mechanism recommends the exercises that students are most likely to answer incorrectly, thus enhancing students' understanding of the knowledge points with lower levels of knowledge mastery proficiency.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, an exercise recommendation system based on knowledge tracing and graph neural network is proposed. The system consists of three modules: the exercise knowledge point tagging module, knowledge tracing module, and exercise recommendation module. The exercise knowledge point tagging module can label the exercises' knowledge points. The marked exercises are adopted as the input data for knowledge tracing and recommendation. The knowledge tracing module tracks the students' question records, thereby tracking the students' knowledge status. The exercise recommendation module combines students' knowledge status and exercise label information to perform exercise recommendations.

In the knowledge point labeling module, a graph convolutional neural network is trained for encoding the relation among knowledge labels and generate labeling classifiers. Bi-LSTM is applied to encode the exercise text information to transform the exercise text to text representation vector as the label classifier group's input. The classifier finally output a set of label classification results. Experiments show that the model has better classification performance for high school math exercise problems than existing baseline models.

In the knowledge tracing module, an improved model for the DKVMN is proposed, which transforms the memory structure of DKVMN with a graph network structure so that the correlations among the memory slots can be represented. Besides, the model introduces the user's answering behavior features as additional inputs for knowledge tracing, thus enhancing representing capability to the model. In the experimental phase, the behavioral features that are most relevant to the output features are selected as inputs by performing data analysis for the dataset data. In comparison with the baseline model, the proposed model achieves the best performance.

In the exercise recommendation module, a two-stage Matching-Ranking based mathematical exercise recommendation model is proposed. The Matching model in the first stage is a multiplexed exercise recommendation item matching model, which applied multiple sub-matching strategies based on collaborative filtering, popularity, user preferences for generating exercise recommendation candidate sets separately, and then these candidate sets are weighted and fused in the fusion stage to form a final fused exercise recommendation candidate set. In the ranking phase, each exercise in the set of exercise candidates obtained in the Matching phase is input to the knowledge tracing exercise for correctness prediction, and the most error-prone exercises are used as the recommendation item with the highest priority. The recommended set of exercises is generated according to this strategy.

5.2 Future Work

There are still some shortcomings in this study, and the proposed method and model can be improved in the future work.

For the exercise knowledge labeling model, the popular pre-trained models (such as BERT, GPT) which can provide a deeper semantic understanding of the exercise text can be applied for word embedding or the text classification. Alternatively, a hierarchical labeling model can be introduced to solve the hierarchical label classification problem.

For the knowledge tracing model, the Multi-head attention mechanism can be applied to the knowledge point relation representation to generate trainable adjacency relation, which improves model flexibility. Also, more feature information can be introduced to enhance the higher-order feature characterization capability of the model.

For the exercise recommendation model, neural network models can adaptively adjust each matching strategy's hyperparameters to reduce manual tuning costs. The recommendation ranking model can add more features to enhance the comprehensiveness of the model recommendations.

References

[1] Yunxiao Chen, Xiaoou Li, Jingchen Liu, and Zhiliang Ying. Recommendation system for adaptive learning. *Applied psychological measurement*, 42(1):24–41, 2018.

[2] Alireza Soltani and Alicia Izquierdo. Adaptive learning under expected and unexpected uncertainty. *Nature Reviews Neuroscience*, 20(10):635–644, 2019.

[3] Fei Wang, Qi Liu, Enhong Chen, Zhenya Huang, Yuying Chen, Yu Yin, Zai Huang, and Shijin Wang. Neural cognitive diagnosis for intelligent education systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6153–6161, 2020.

[4] Hung-Yu Huang. Multilevel cognitive diagnosis models for assessing changes in latent attributes. *Journal of Educational Measurement*, 54(4):440–480, 2017.

[5] Benidiktus Tanujaya, Jeinne Mumu, and Gaguk Margono. The relationship between higher order thinking skills and academic performance of student in mathematics instruction. *International Education Studies*, 10(11):78–85, 2017.

[6] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2019.

[7] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[8] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[9] Konstantinos Pliakos, Seang-Hwane Joo, Jung Yeon Park, Frederik Cornillie, Celine Vens, and Wim Van den Noortgate. Integrating machine learning into item response theory for addressing the cold start problem in adaptive learning systems. *Computers & Education*, 137:91–103, 2019.

[10] Hung-Yu Huang. Utilizing response times in cognitive diagnostic computerized adaptive testing under the higher-order deterministic input, noisy ‘and’ gate model. *British Journal of Mathematical and Statistical Psychology*, 73(1):109–141, 2020.

[11] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*, pages 171–180. Springer, 2013.

[12] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. *Advances in neural information processing systems*, 28:505–513, 2015.

[13] Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo. Graph-based knowledge tracing: modeling student proficiency using graph neural network. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 156–163. IEEE, 2019.

[14] Zhenya Huang, Qi Liu, Chengxiang Zhai, Yu Yin, Enhong Chen, Weibo Gao, and Guoping Hu. Exploring multi-objective exercise recommendations in online education systems. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1261–1270, 2019.

[15] Wang Juan, Lan Yue-xin, and Wu Chun-ying. Survey of recommendation based on collaborative filtering. *Journal of Physics: Conference Series*, 1314:012078, oct 2019.

[16] Soojung Lee. Improving jaccard index for measuring similarity in collaborative filtering. pages 799–806, 03 2017.

[17] Luong Vuong Nguyen, Min-Sung Hong, Jason J. Jung, and Bong-Soo Sohn. Cognitive similarity-based collaborative filtering recommendation system. *Applied Sciences*, 10(12), 2020.

[18] Jiangzhou Deng, Yong Wang, Junpeng Guo, Yongheng Deng, Jerry Gao, and Younghee Park. A similarity measure based on kullback-leibler divergence for collaborative filtering in sparse data. *Journal of Information Science*, 45(5):656–675, 2019.

[19] Wei Fu, Jun Liu, and Yirong Lai. Collaborative filtering recommendation algorithm towards intelligent community. *Discrete & Continuous Dynamical Systems-S*, 12(4&5):811, 2019.

[20] Prem Gopalan, Laurent Charlin, David M Blei, et al. Content-based recommendations with poisson factorization. In *NIPS*, volume 14, pages 3176–3184, 2014.

[21] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[22] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.

[23] Weiwei Liu, Xiaobo Shen, Haobo Wang, and Ivor W. Tsang. The emerging trends of multi-label learning, 2020.

[24] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5177–5186, 2019.

[25] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 207–212, 2016.

[26] Kenneth Ward Church. Word2vec. *Natural Language Engineering*, 23(1):155–162, 2017.

[27] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. Revisiting pre-trained models for Chinese natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 657–668, Online, November 2020. Association for Computational Linguistics.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[29] Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. A survey on long short-term memory networks for time series prediction. In *2020 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering (ICME), Gulf of Naples*, 2020.

[30] Gang Liu and Jiabao Guo. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338, 2019.

[31] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, Sang-ug Kang, and Jong Wook Kim. Bi-lstm model to increase accuracy in text classification: combining word2vec cnn and attention mechanism. *Applied Sciences*, 10(17):5841, 2020.

[32] Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models, 2020.

[33] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4438–4446, 2017.

[34] Ming Sun, Yuchen Yuan, Feng Zhou, and Errui Ding. Multi-attention multi-class constraint for fine-grained image recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 805–821, 2018.

[35] Dichao Hu. An introductory survey on attention mechanisms in nlp problems. In *Proceedings of SAI Intelligent Systems Conference*, pages 432–448. Springer, 2019.

[36] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*, 2015.

[37] Le Wu, Peijie Sun, Richang Hong, Yanjie Fu, Xiting Wang, and Meng Wang. Socialgcn: an efficient graph convolutional network based model for social recommendation. *arXiv preprint arXiv:1811.02815*, 2018.

[38] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[39] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[40] Suprianto Panjaitan, Muhammad Amin, Sri Lindawati, Ronal Watrianthos, Hengki Tamando Sihotang, Bosker Sinaga, et al. Implementation of apriori algorithm for analysis of consumer purchase patterns. In *Journal of Physics: Conference Series*, volume 1255, page 012057. IOP Publishing, 2019.

[41] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics.

[42] Bao Guo, Chunxia Zhang, Junmin Liu, and Xiaoyi Ma. Improving text classification with weighted word embeddings via a multi-channel textcnn model. *Neurocomputing*, 363:366–374, 2019.

[43] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[44] José González-Brenes, Yun Huang, and Peter Brusilovsky. General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. In *The 7th international conference on educational data mining*, pages 84–91. University of Pittsburgh, 2014.

[45] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

[46] Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*, pages 765–774, 2017.

[47] Chia-Yi Chiu, Yan Sun, and Yanhong Bian. Cognitive diagnosis for small educational programs: The general nonparametric classification method. *psychometrika*, 83(2):355–375, 2018.

[48] Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. EDM 2019 - Proceedings of the 12th International Conference on Educational Data Mining, pages 384–389. International Educational Data Mining Society, 2019.

[49] Y. Lee, Youngduck Choi, Jung hyun Cho, Alexander R. Fabbri, Hyunbin Loh, Chanyou Hwang, Yongku Lee, Sang-Wook Kim, and Dragomir R. Radev. Creating a neural pedagogical agent by jointly learning to review and assess. *ArXiv*, abs/1906.10910, 2019.

[50] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[51] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[52] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[53] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[54] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 255–262, 2016.

[55] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1725–1731. AAAI Press, 2017.