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

Wangzhihui Mei

Supervisor: Zhifeng Wang

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

Central China Normal University Wollongong Joint Institute
Central China Normal University
March 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.

Wangzhihui Mei
March 2021

Abstract

After 2010, artificial intelligence technology has gradually become a popular research topic in computer technology. In particular, the advent of AlphaGo has aroused great concern in the industry for the prospect of artificial intelligence. This has brought about the industry's explosive development and raised a large number of research topics. In artificial intelligence-related research, various algorithm innovations, theoretical breakthroughs, and model applications are emerging one after another, laying the foundation for various industries' intelligence. The application of artificial intelligence technology in the field of education has also given birth to the emergence of the concept of intelligent education. Among them, adaptive learning is one of the popular application fields in intelligent education[1]. Adaptive learning models generally track students' learning status by combining extensive data analysis of massive student group learning data and precise data analysis of target student individual data, targeting on personalizing the learning path according to the individual characteristics of the students and the proficiency of knowledge mastery[2]. Adaptive learning technology can use automated machine learning algorithms to complete student evaluation and teaching plans that required much manual labor in the past, which can systematically alleviate the current scarcity and uneven distribution of domestic educational resources and reduce the burden on education practitioners and students. It also has excellent development prospects and commercial value. More and more artificial intelligence research teams and intelligent education technology companies on the market focus on developing and applying adaptive learning tools. Some smart educational technology companies have used adaptive learning as the core function or main selling point of their products. Adaptive educational technology can comprehensively analyze students' individual-level learning ability, knowledge proficiency, group-level popular learning resources, error-prone questions, etc. The most suitable learning path and learning resources such as exercises, materials, and knowledge points can be pushed to the student. According to the students ' knowledge status, the system automatically adjusts the knowledge focus of the pushed learning resources to prevent the repetitive practice of already mastered knowledge points or lack of practice of unmastered knowledge points.

On the one hand, teachers can analyze the whole class's knowledge mastery proficiency based on the data or visual charts output by the system to create a learning status assessment report for each student and adaptively adjust the overall teaching plan. On the other hand, students can use the system to analyze their knowledge weaknesses and target appropriate exercises. Thus, adaptive learning is one of the potentially feasible solutions to the "automatic assessment of students" knowledge mastery status and instructional program generation" in online education.

The purpose of this paper is to propose a knowledge-tracing-based model for recommending high school mathematics exercises, using the subject of high school mathematics as the primary research context. In high school mathematics, practice exercises are the principal means for students to improve their learning ability. However, in the current high school mathematics teaching, teachers or students need to find suitable exercises to practice from a vast library of exercises, which are often too large, highly repetitive, and confusingly organized. There are many low-quality unlabeled exercises in the exercise bank, which need to be manually labeled with knowledge points. Some students study through excessive exercises tactic, but this is less efficient and often results in repetition of familiar knowledge and avoidance of unfamiliar knowledge. To improve the effectiveness of students' exercises and thus enhance the proficiency and comprehensiveness of knowledge acquisition, the experienced teaching staff is needed to conduct an analysis of students' knowledge status and select appropriate exercises from the exercise bank for a recommendation. The method is uneconomical and inefficient because of its high manual workload, its reliance on expert a priori knowledge, and its inclusion of a large amount of repetitive work. Besides, the traditional exercise recommendation takes the student group as the minimum granularity. However, it does not perform recommendations based on the knowledge mastery proficiency of specific students, which ignores the problem that different students have different learning abilities, so the recommendation is less fine-grained and ineffective for most students. The method is uneconomical and inefficient because of its high manual workload, its reliance on expert a priori knowledge, and its inclusion of a large amount of repetitive work. Also, the traditional exercise recommendation takes the student group as the minimum granularity while does not recommend for the knowledge mastery proficiency of specific students, which ignores the problem that different students have different learning abilities, so the recommendation is less fine-grained and ineffective for most students. To improve the problems of traditional exercise recommendation methods, knowledge tracing techniques can be applied to track students' learning and thus target automated exercise recommendations. This thesis aims to design an exercise recommendation system based on knowledge point labeling, knowledge

tracing, and resource recommendation techniques and thus introduce an intelligent adaptive learning solution in terms of exercise recommendation.
The proposed recommendation system for high school mathematics exercises consists of three modules: the exercise knowledge point labeling module, the knowledge tracing module, and the recommendation module. The exercise knowledge point labeling module's function is to perform knowledge point labeling for exercises without knowledge point labeling, thus replacing the traditional manual knowledge point labeling with an automated form. The knowledge point labeling is the pre-work of the exercise recommendation, and the knowledge labeled exercises can be used as the data source of the exercise recommendation system. The knowledge tracing module calculates students' knowledge proficiency state vector, which represents students' mastery of subject knowledge points, concepts, and skills, by tracking students' exercise records. Knowledge tracing is the core part of the system. In the final exercise recommendation module, there are two stages of matching and sorting; the former is applied to the exercise database to apply a variety of matching strategies to quickly filter the exercises and generate a collection of recommended candidate exercises, and the latter inputs the collection into the knowledge tracing system in the sorting stage for refined recommendation sorting to generate the final recommendation results.

- Chapter 2 proposes a multi-knowledge point labeling method for high school mathematics exercises based on bidirectional LSTM and graph neural network. The exercise knowledge point labeling module contains two sub-modules: exercise text mining and multi-knowledge point label classification. Since most of the corpus exercises contain only unstructured data such as textual information, this paper focuses on knowledge point extraction utilizing exercise text mining. It applies a bidirectional LSTM network with an attention mechanism to perform exercise text mining. The exercises are firstly pre-processed by word separation, cleaning, regularization, and other preprocessing steps to obtain word sequences while filtering out a large amount of interference of irrelevant information. Next, the pre-trained BERT vector generation approach is used as the learning word embedding vector instead of the simple one-hot encoding approach, preventing the dimensional disaster caused by the sparse input word vector matrix. Moreover, the hidden dependencies between word vectors can also be characterized by embedding learning, which is conducive to inter-knowledge point dependencies. After that, text information extraction by a bidirectional LSTM model can better solve the problem of long-range dependency of contextual elements in the text. Also, to capture inter-knowledge point dependencies on the classification model, a multi-label knowledge point labeling model based on graph convolutional network (GCN) is proposed in this paper, where a graph embedding of knowledge

points represents each label, and the label graph is mapped into a set of intrinsically dependent knowledge point classifiers after several rounds of iterative learning. Subsequently, the word vector of the exercise text extracted from the previous sub-network is fed into the set of knowledge point classifiers to derive a multi-knowledge point prediction probability vector, thus realizing the multi-knowledge point labeling task. In the experimental phase, the proposed method in this thesis is compared with a series of baseline models by conducting experiments on a self-made high school mathematics exercise dataset, and a series of multi-label classification metrics are used to compare and evaluate the model performance. The experimental results show that the method achieves more superior performance on the more complex sets of exercises with more complex knowledge point relationships.s method has achieved superior performance on the problem sets with more complex knowledge point relationships.

- Chapter 3 proposes a knowledge tracing model based on Transformer architecture with graph attention network (GAT) embedding. The model learns the complex correlations between exercises at the knowledge point level through graph attention networks and optimizes the traditional knowledge tracing model for the shortcoming of insufficient characterization of complex correlations between knowledge points between exercises. It solves the following problems, (1) The traditional model models knowledge points as mutually independent relationships or simplified probabilistic dependencies while ignoring the complex graph-like relationships among knowledge points, thus performing poorly for data with complex knowledge point dependencies. (2) The traditional model cannot output the students' knowledge state, but only the probability of correct answer for the next exercise, making it difficult to combine with the recommendation model to recommend exercises. The proposed model combines the powerful representational learning capability of graph attention networks for data and graphs in non-Euclidean space and the remote-dependent attention mechanism modeling capability of the Transformer model for serialized exercise data, which has better performance in handling longer exercise records and complex exercise datasets with knowledge relationships. In the experimental phase, the model's performance proposed in this thesis is compared with the baseline model by conducting experiments on several publicly available datasets in the knowledge tracing domain. The experimental results show that the present model achieves better or comparable results in terms of evaluation parameters relative to most publicly available datasets.

- Chapter 4 presents a mathematical exercise recommendation model based on 2 phases of Matching-Ranking. The first stage is the matching model, which is a hybrid model

based on multiple matching strategies with two processes: multiplex-matching and merging. In the process of multiplex-matching, multiple matching strategies such as collaborative filtering, popularity, user preferences are used to generate several subsets of exercise recommendation candidates separately. Then, in the process of merging, these candidate subsets are merged by weighted ranking to form a final set of exercise recommendation candidates. The second stage is a knowledge-tracing-based recommendation item ranking model, in which each exercise in the set of exercise candidates obtained in the previous stage is input to the knowledge-tracing model proposed in the previous chapter for correctness prediction, and the one with the most error-prone exercises is used as the recommendation item with the highest priority. The model mainly solves the problem that traditional models often require user-initiated ratings for recommendation model construction and cannot make efficient recommendations based on the user's knowledge mastery proficiency state. After the performance test on the public dataset and the control experiment with the baseline model, the proposed model has a better performance in tracking the students' weak knowledge mastery and can recommend the appropriate exercises based on the mastery proficiency status obtained from the tracking.

This paper analyzes the system's requirements, rationalizes the entire recommendation system into multiple modules, and designs different neural network models and algorithms for each module to achieve and optimize the above three modules. It has both algorithm design and experimental verification methods—a certain degree of innovation. After experimental verification, it has better performance than similar models.

Keywords: Graph Neural Network, Knowledge Point Labeling, knowledge tracing, Recommended Exercises

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，也提出了大量的研究课题。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 [1]。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 [2]。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。它也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工具的研发和应用，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” 问题潜在可行解决方案之一。

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的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。知识点标注是习题推荐的前置工作，经过知识标注的习题可以作为习题推荐系统的数据源。知识追踪模块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、概念和技能的掌握度的表征。知识追踪是整个系统的核心部分。在最后的习题推荐模块，具有召回和排序两个阶段，前者应用于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，后者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。

- 第二章提出了一种基于双向 LSTM 与图神经网络嵌入学习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，习题首先经过分词、清洗、正则化等预处理步骤，得到词序列的同时过滤掉大量的无关信息的干扰。接下来，通过预训练 BERT 模型生成的方式而非简单的 one-hot 编码方式来作为学习词嵌入向量，这样做可以防止输入词向量矩阵稀疏带来的维数灾难等问题。而且，通过嵌入学习的方式，也可以表征词向量间的隐藏依赖关系，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。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网络（GCN）的多标签知识点标注模型，每个标签由知识点的图嵌入表示，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，将前一个子网络提取的习题文本词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在知识点关系较为复杂的习题集上取得了更加优越的性能。

- 第三章提出了基于图注意力网络（GAT）和 Transformer 架构的知识追踪模型。该模型通过图注意力网络进行图嵌入学习的方式来表征学习习题间在知识点层面上的复杂关联关系，针对传统的知识追踪模型对于习题间知识点复杂关联关系表征不足的缺陷进行了优化。它解决了如下问题，（1）传统模型将知识点建模为相互独立的关系或者简化的的概率依赖关系，而忽略了知识点间复杂的图状关系，从而对于知识点依赖关系复杂的数据表现不佳。（2）传统模型无法输出学生的知识状态，而只能输出对于下一道习题的回答正确概率，从而难以结合推荐模型进行习题推荐。本文提出的模型结合图注意力网络对于非欧式空间的数据和图的强大表征学习能离和 Transformer 模型对于序列化习题练习数据的远程依赖的注意力机制建模能力，在处理较长的习题练习记录和知识关系复杂的习题数据集上具有更佳的性能。实验阶段，通过在若干个知识追踪领域公开数据集上进行实验，将本论文提出的模型与基准模型进行性能对比。实验结果显示，在公开数据集上，本模型相对于大多数模型在评估参数上都取得了较优或者相当的结果。

- 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每个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的作为优先级最高的推荐项。该模型主要解决的是传统模型往往需要用户主动评分来进行推荐模型构建，而无法基于用户的知识掌握熟练度状态进行高效推荐的问题。经过在公开数据集上的性能测试和与 baseline 模型的对照实验，提出的模型在对于学生的掌握薄弱知识的追踪性能较为优越，并能依据追踪得到的的知识掌握熟练度状态推荐合适的习题。

本文通过分析系统的需求，将整个推荐系统合理化分为多个模块，并针对各个模块设计了不同的神经网络模型和算法来实现和优化上述三个模块，在算法设计和实验验证方法方面都具有一定的创新性。经过实验验证，具备相对于同类模型更好的性能。

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. Zhifeng 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 your friendship and support!

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 the structure of industry chains, and increasing the efficiency of information utilization. As a traditional service industry, the education industry also has considerable room for an intelligent revolution. In traditional education models, student groups are often divided 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, which significantly impact 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. 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. 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 paper 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

The first part of this paper is a graph neural network, and natural language processing (NLP) 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 paper 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 [3]. 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 very 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, etc. [4] By analyzing the knowledge points, a network of knowledge point associations can be established to facilitate subsequent knowledge mastery status tracking.

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 paper 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 was proposed in 2009 by Franco et al. The model is based on the theory of immobile points and aims to obtain the hidden state of each node. However, when the number of stacks is too many, the state convergence tends to be too smooth and makes it 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 paper, graph convolutional neural networks (GCN) and graph attentional neural networks (GAT) are deployed. GCN is a model proposed by Kipf in 2016 for semi-supervised classification [5], 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 [6]. 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 (KT) is a technology that models student knowledge acquisition based on past answer records and results. It is a typical 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) [7] 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 [8]. Besides, Bayesian knowledge tracing (BKT) is a more widely used knowledge tracing model based

on probabilistic graphical modeling [9], 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 [10], 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. And the Dynamic Key-Value Memory Networks for Knowledge Tracing (DKVMN) model proposed in 2017 borrows the idea of memory-enhanced networks, which can store students' mastery levels of knowledge concepts using a key 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 thus 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 [11], 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 the 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 of adopting the sea tactics is low, and the results are slow. Therefore, a system for adaptive exercise recommendation based on students' knowledge status is necessary and important [12].

There are several types of algorithms in recommendation systems: collaborative filtering, content-based recommendation models, etc. 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 [13]. 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 [14], cosine similarity [15], related similarity [16], etc. Collaborative filtering often has a "cold boot" problem, 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 [17].

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 Chinese math exercises, building a knowledge point annotation model, and performance testing of the model. In the second part, the labeled knowledge point exercises are used as the input of the knowledge tracing model, which learns the knowledge graph embedding representation of the exercises through graphical neural networks, designs a knowledge tracing model based on the Transformer structure 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 rec-

ommendation model based on two stages of matching-ranking is designed first to filter the exercises to filter the candidate recommended exercises, then input the knowledge tracing model to predict the probability of correct answers, and output a final list of recommended exercises according to the ranking rule that the higher the error rate is, the higher the priority is.

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 paper 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 annotation, knowledge tracing, and exercise recommendation.

Chapter 2 of this paper focuses on the knowledge point annotation of 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 according to their current knowledge mastery. This chapter's algorithmic model is divided into two parts: exercise text information extraction and label annotation. In the experimental part, the model's effectiveness is verified by conducting performance comparison experiments with several baseline models.

Chapter 3 of this thesis proposes a knowledge tracing model based on a graph neural network. The model is divided into three parts: exercise-knowledge point relationship embedding learning, knowledge state encoding, and answer prediction decoding. The article first introduces the design ideas and related technologies theoretically and then compares the benchmark model with the public data set in the experimental part to verify the model's effectiveness and evaluate the performance of the model.

Chapter 4 of this thesis proposes a recommendation system model based on 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 correct probabilities of the exercises are collected and prioritized to obtain a list of recommended exercises.

Chapter 5 of this paper 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

All aspects of the recommendation system, including data collection, data mining, and data recommendation, are predicated on establishing a high-quality data source. In this thesis's study topic, establishing a standardized and structured exercise database is the critical first step to build a test recommendation system. The construction of an exercise corpus is also a complex and challenging task, requiring comprehensive consideration of all aspects of the exercise data and adding enough additional information to the exercises for subsequent data mining. 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 (textbooks), 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 threshold 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 central 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 prob-

lem, 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.

In this chapter, the knowledge point labels are mined for topic recommendation and analysis reports. It requires refining the knowledge point association of the topic, i.e., several corresponding knowledge point tags for the topic, where 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. In this definition, 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.

2.2 Proposed Model

2.2.1 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 tracking 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, the manual labeling has problems such as 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 [18–20]. 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 [21]. 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.

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

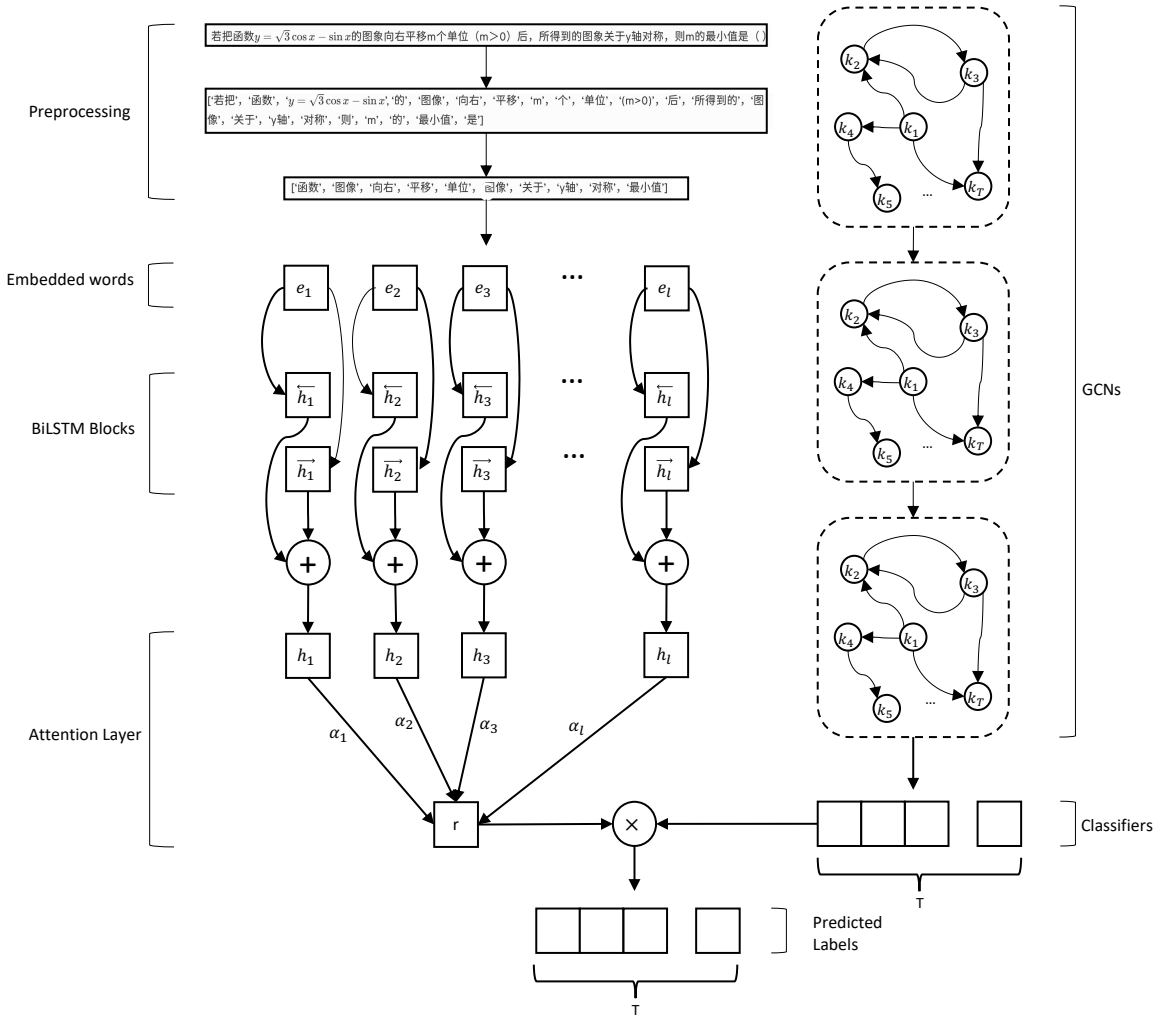


Fig. 2.1 Structure of the knowledge labeling model

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 [22], 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 computed with the exercise information representation vector outputted by the first part to output a result vector representing each knowledge point's labeling probability.

2.2.2 The Exercise Description Text Mining

This section is based on the Attention based Bi-LSTM, whose architecture design is shown in Fig.2.8.

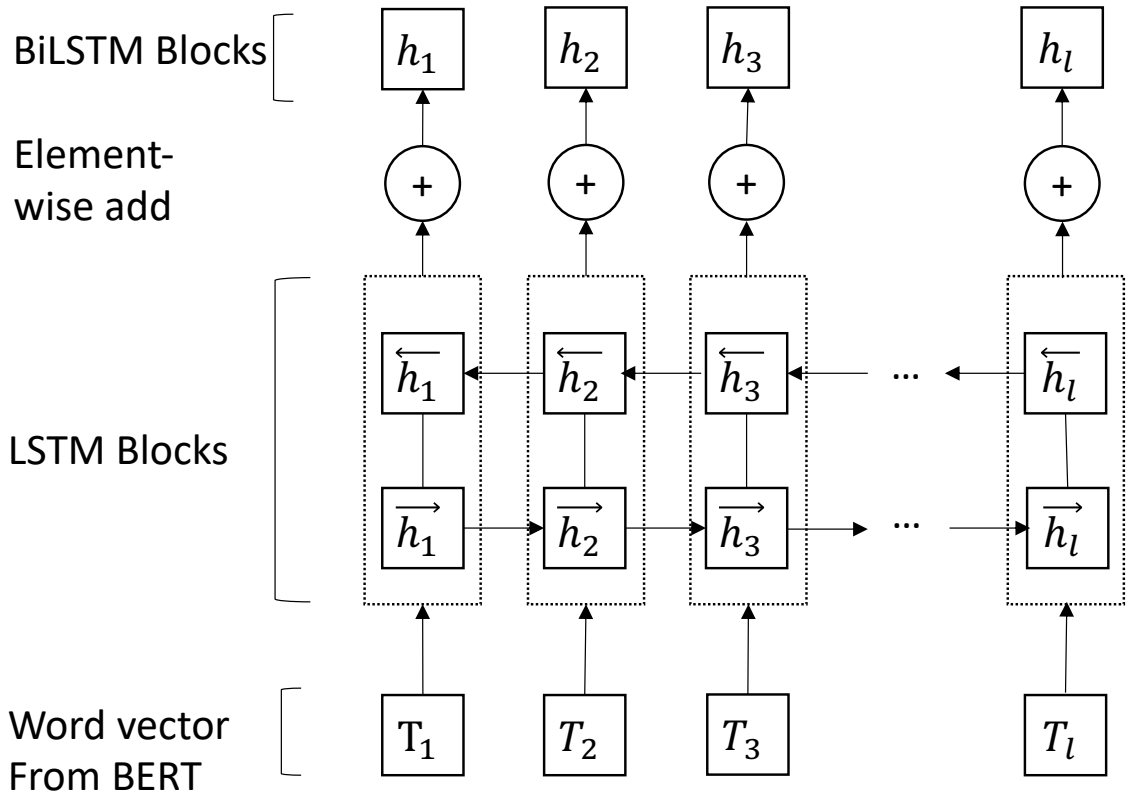


Fig. 2.2 Structure of attention based Bi-LSTM

The model includes 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.2.3 shows an example of preprocessing of an exercise description text.

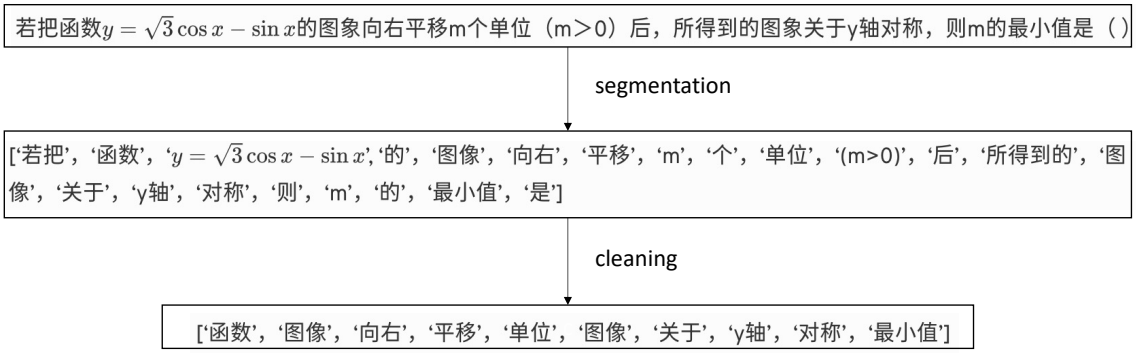


Fig. 2.3 Example of preprocessing

- Segmentation: Compared to other phonetic languages such as English and Spanish, Chinese language preprocessing requires an extra step of word separation. There is no natural separation between words in Chinese sentences. Therefore artificial segmentation of sentences according to word meanings is required. Choosing a suitable Chinese word segmentation algorithm can lay the foundation for the subsequent processing and improve the overall model performance. At present, the Chinese word segmentation is mainly divided into two types of algorithms: rule-based lexical matching and statistical model-based. Compared with the former, the latter has better generalization ability and learning ability. It performs better for word separation beyond the a priori rules, such as ambiguous words and unregistered words. In this model, the currently popular jieba subword is used to implement a word graph scan based on the Trie book structure, and algorithms such as stage planning and HMM model are used to find the maximum subword grouping and merging based on word frequency to achieve the identification of future logged-in words. Users can also customize stop words and user dictionaries from achieving proper noun recognition.

- 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 a large number of 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, symbols, etc. 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 too many dimensional 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 [23], 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 training sample generation phase of BERT, these combinations may be masked separately, resulting in ambiguity and causing training performance degradation. Therefore, this thesis applies Whole Word Mask processing (WWM), which was proposed by Cui et al. in 2019 [24]. 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.2.4.

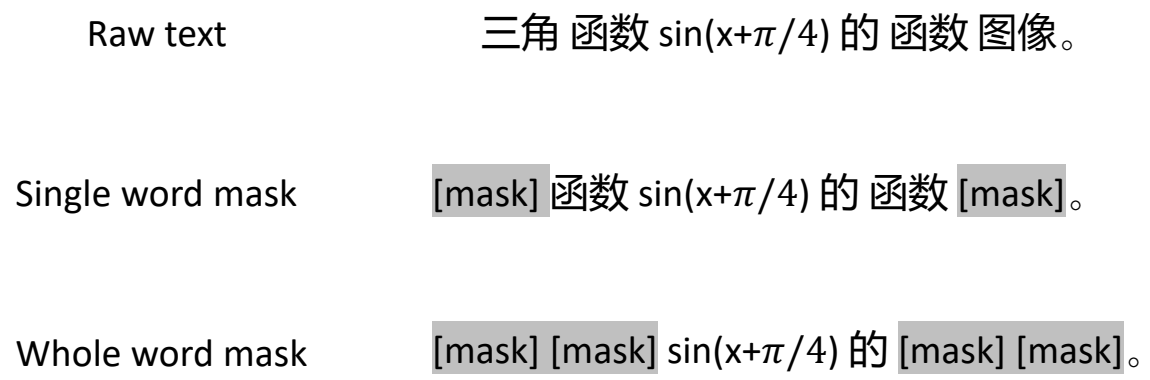


Fig. 2.4 Whole word mask

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.2.5.

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.fig:ch2-rnn-model, 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 2.1:

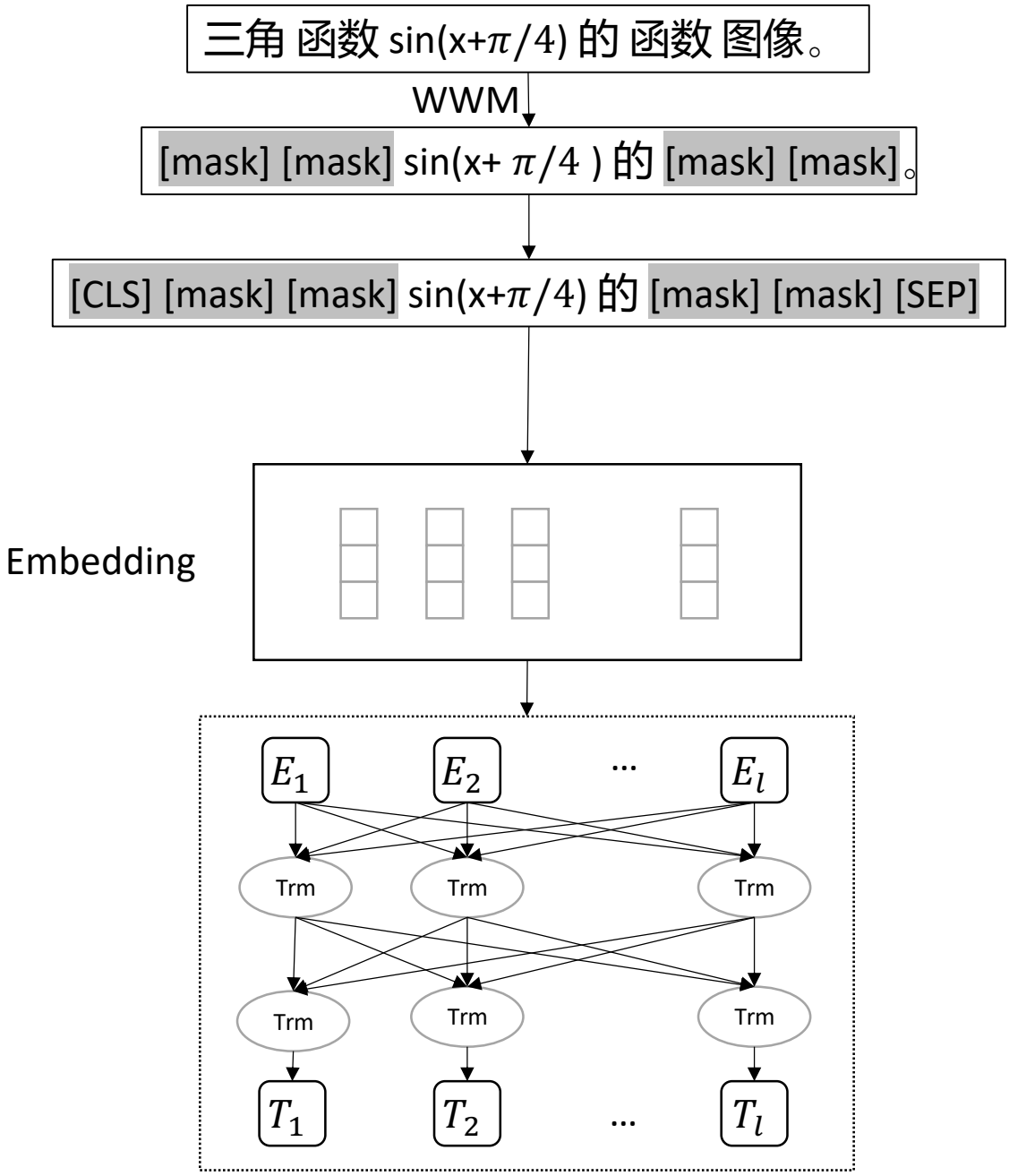


Fig. 2.5 BERT-based embedding

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

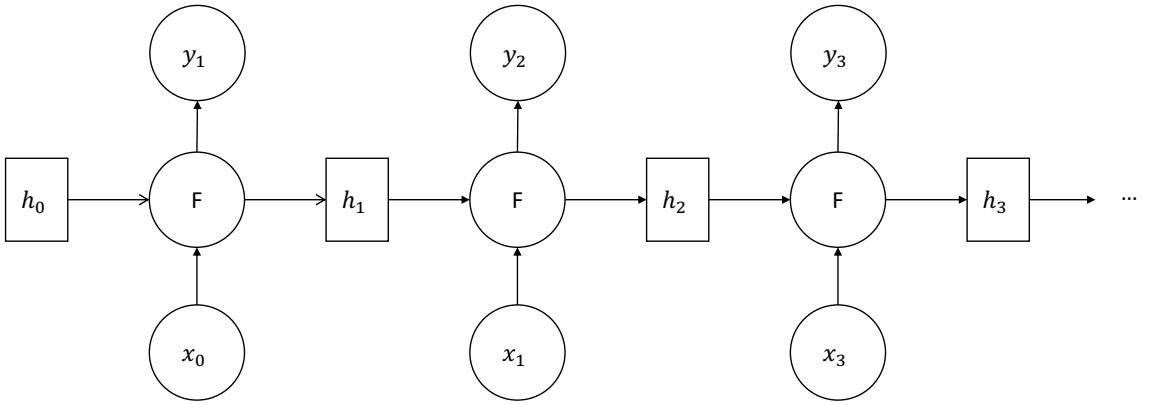


Fig. 2.6 Naive RNN unit

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) [25], 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.2.7, 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) [26] 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. The Bi-LSTM output can be obtained by inputting the positive sequence and the reverse sequence input sequence into 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

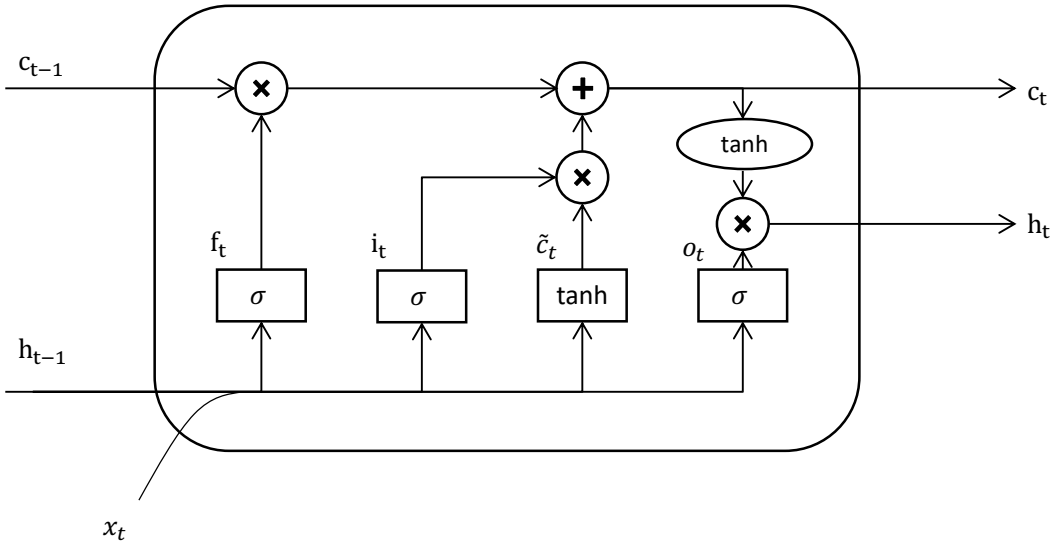


Fig. 2.7 Structure of LSTM unit

output of Bi-LSTM is:

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

The Bi-LSTM Structure is like Fig.2.8.

Here, the Bi-LSTM output a bidirectional sequence as the concatenation of positive-order and reverse-order LSTM output sequence.

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. The proposed Attention model proposed by Bahdanau et al. is based on [27]. 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 remarkable results in different tasks such as image vision [28, 29], language mining [30], and voice recognition [31] 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 mech-

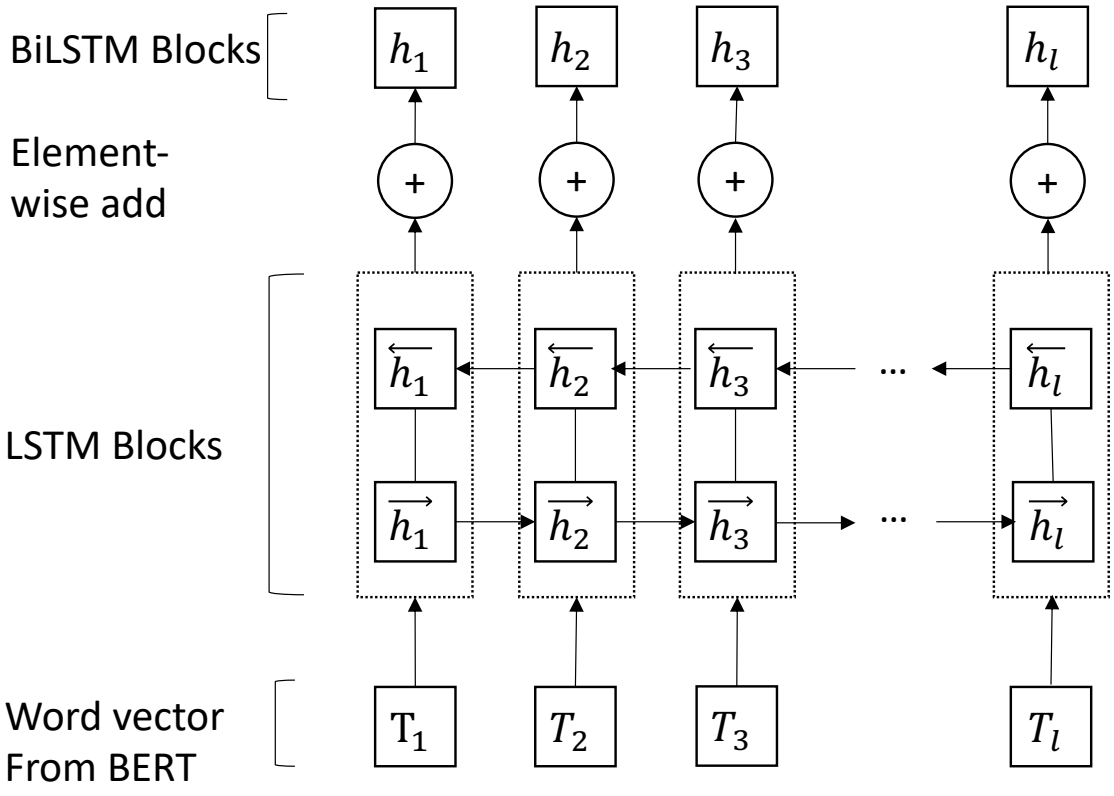


Fig. 2.8 Structure of Bi-LSTM

anism. This method is widely used in various types of deep learning tasks such as NLP [30], image classification [28, 29], and speech recognition [31], and has achieved remarkable results.

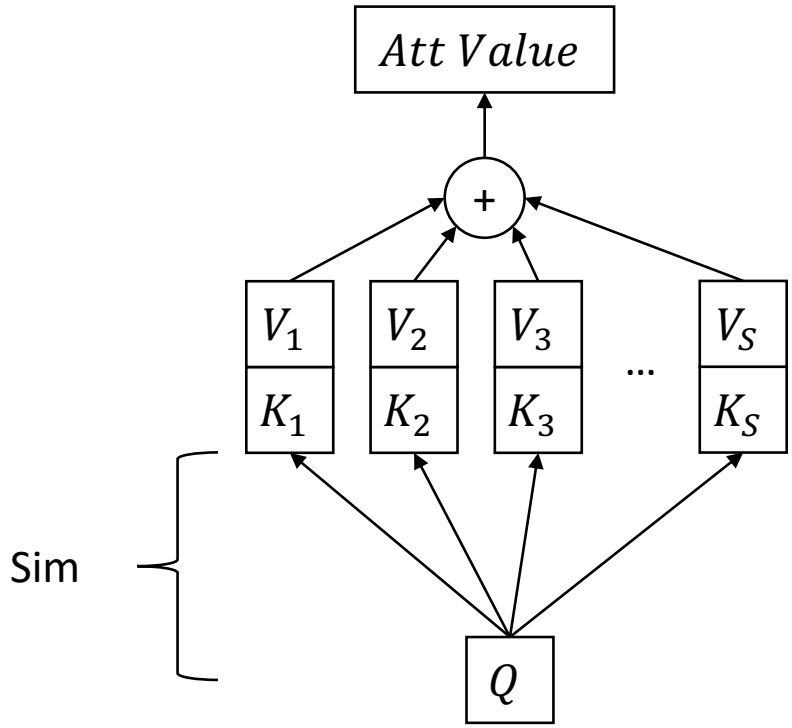


Fig. 2.9 The essential idea of attention mechanism

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.2.9. Then a weighted sum is performed to obtain the final Attention value. It can be expressed as 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 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 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.

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

2.2.3 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.2.10.

Graph Convolutinal Network

In the real world, many important data sets generate connections as networks that form graph-like structures. Several papers reviewed this problem and attempted to generalize neural networks and apply them to arbitrary graph-structured data [32, 33]. The GCN [5] is used

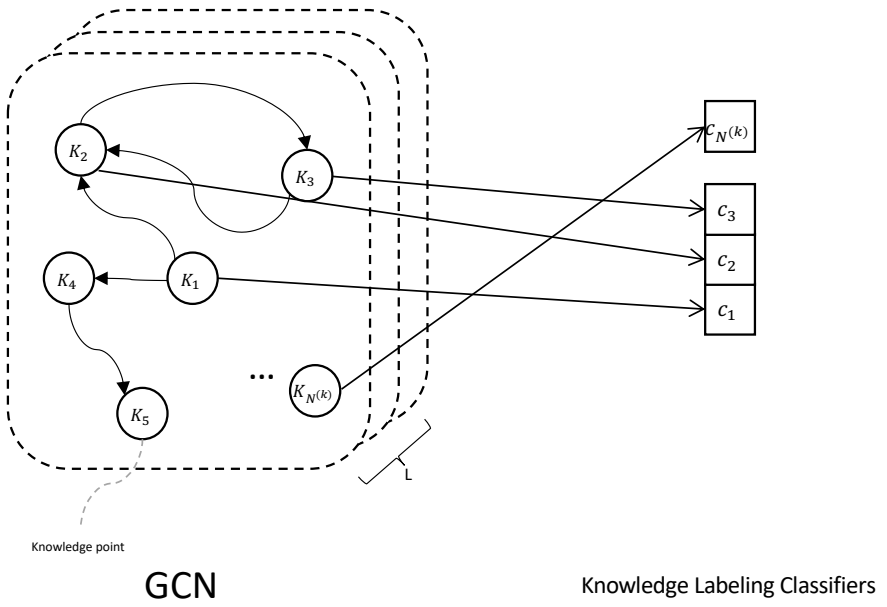


Fig. 2.10 The overview of structure

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.

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 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 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 [34]. 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.2.11.

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

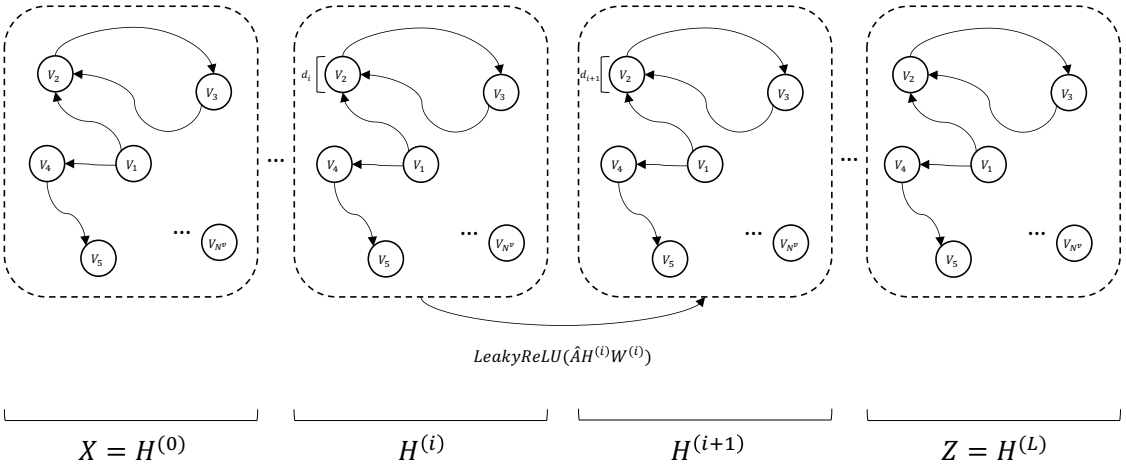


Fig. 2.11 The GCN training process

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. 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 $\mathscr{L}_i$ appearing, another label $\mathscr{L}_j$ appears, or the con-

ditional probability of the data. Lift represents the probability that the label $\mathscr{L}_i$ is contained at the same time, and the ratio of the probability of occurrence of X population:

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

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 $P_{ij} = P(\mathscr{L}_i, \mathscr{L}_j)/P(\mathscr{L}_j)$, where $P_{ij} = P(\mathscr{L}_i, \mathscr{L}_j)/P(\mathscr{L}_j)$ 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 2.8.

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

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 [35], and calculate the co-relation matrix by calculating the co-occurrence of knowledge points in the exercises.

Classifier generator

In this thesis, a learnable stacked GCN-based model is proposed. The knowledge point labels are represented by knowledge label word embedding. Knowledge point set K =

$\{k_1, k_2, \ldots, k_{N^{(k)}}\}$, 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 $C = [c_1, c_2, \ldots, c_{N^{(k)}}$, 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.2.12.

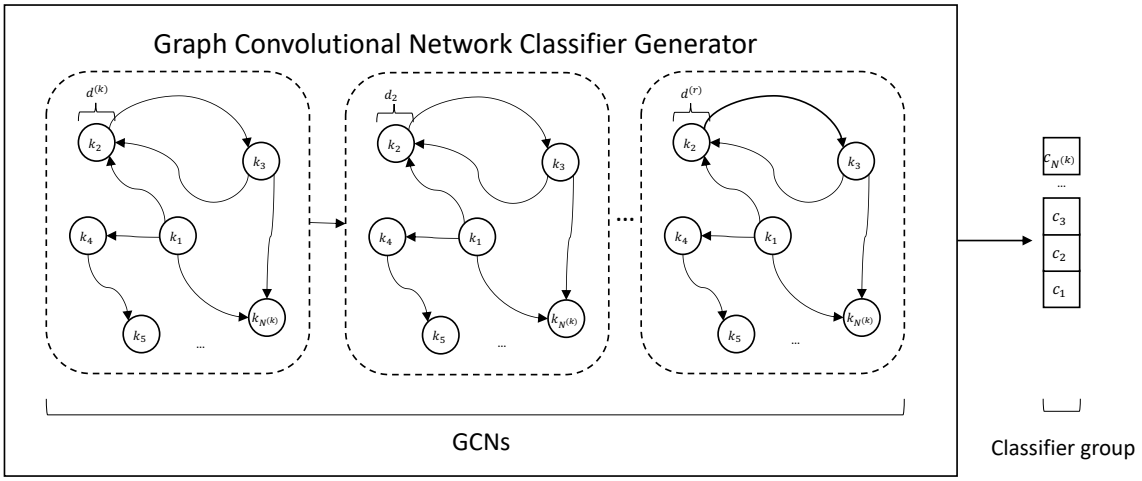


Fig. 2.12 The structure of GCN-based classifier generator

2.2.4 Multi-label Recognition

From the stacked GCN network, the object classifiers $C = \{c_1, c_2, \ldots, c_{N^{(k)}}\}$ 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:

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

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 $\mathbf{L}$ can be written as:

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

2.3 Experiments

This chapter proposes a multi-label labeling model for exercise knowledge points. This section first introduces the data set, then introduces some baseline performance models, and then combines the multi-label labeling to propose a comparative performance indicator evaluation plan. Finally, Give comparison results and analysis.

2.3.1 Dataset

The experimental data comes from the labeled college entrance examination math test questions and simulation questions (including answer analysis) on the online website. The text corpus, such as question stems and answer analysis, are crawled through crawlers. Part of the knowledge point association model can be expressed as the following structure.

The data set has been filtered and manually selected. There are 3374 test questions, including 148 knowledge points, with an average of 1.7 knowledge points. The distribution of several knowledge points is shown in the Fig.2.13.

The distribution of the number of exercise knowledge points in this data set is shown in the Fig.2.14.

2.3.2 Baseline

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.

- Naive Bayes algorithm (NB): predict the labeling probability of a knowledge point based on the prior probability of text combination, and then convert the binary classification problem into a multi-label problem
- Multi-label KNN (ML-KNN): Proposed by Zhang et al. [36], it searches the k instances closest to the instance through the KNN algorithm, counts the number of each category in the k samples, uses the Naive Bayes algorithm to generate prediction outcomes for each label.

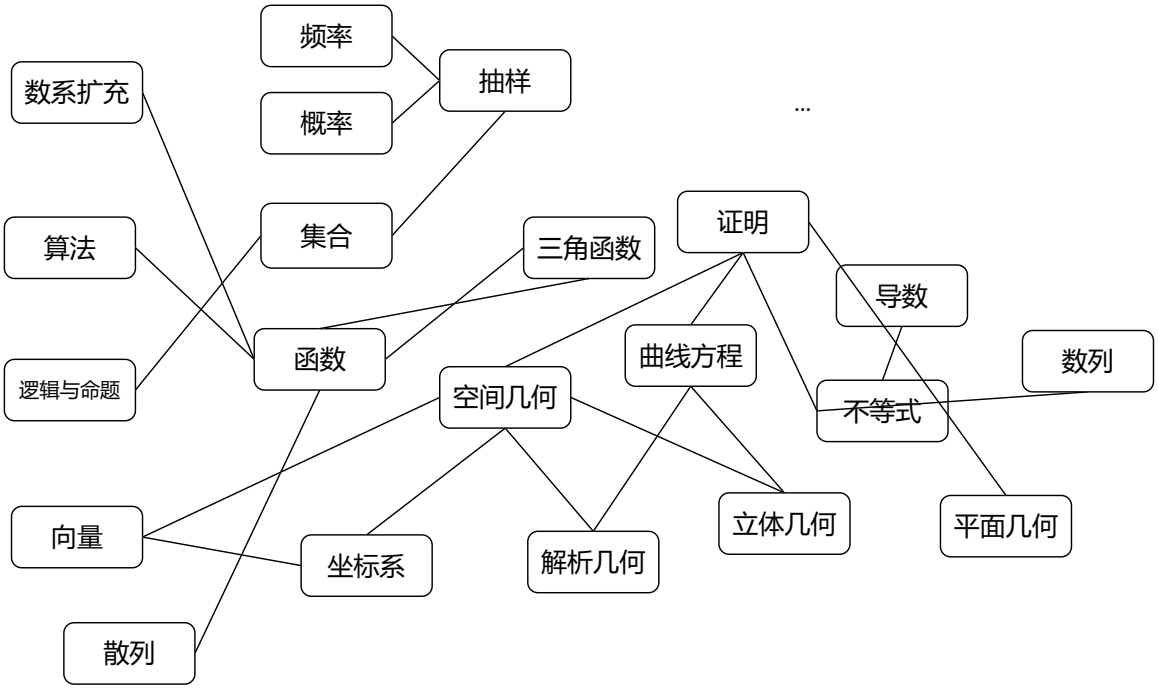
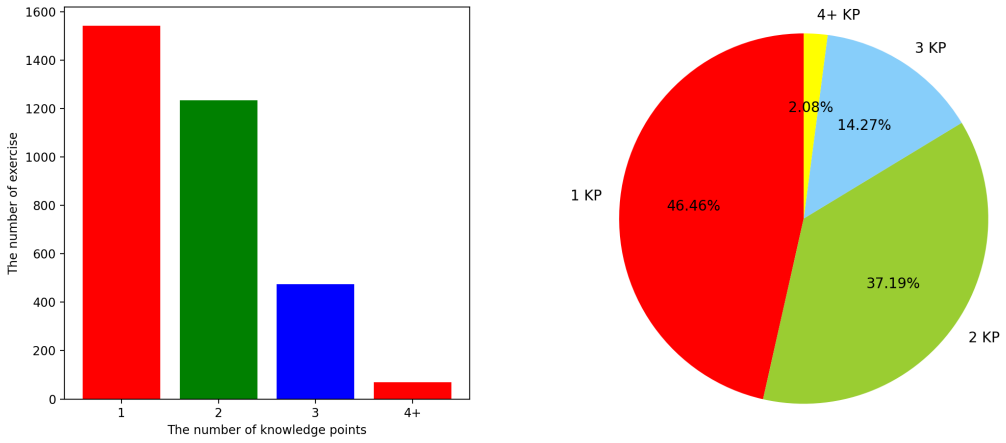


Fig. 2.13 Knowledge points of dataset



(a) Number of exercises (b) Portion of exercises

Fig. 2.14 Distribution of the number of knowledge points of exercise

- CNN+word2vec: Use word2vec to convert the test question text into an embedded vector, extract the test text information through CNN, and then output the multi-label prediction

- CNN+BERT: Use BERT to convert the test question text into an embedded vector, extract the test text information through CNN, and then output the multi-label prediction.

2.3.3 Metrics

Compared with traditional learning problems, it is tough to label multi-label data, and multi-label means huge classification cost [19]. 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 article 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. [19], for the jth label of the ith example in the n examples, shown in the formula 2.11.

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

where x_i and $\mathscr{L}_j$ represent the i-th exercise and j-th label, Y_i and $\tilde{Y}_i$ represent the real labels and predicted labels of the i-th exercise.

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. F1 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 2.12:

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

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 2.13 and 2.14.

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

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

Where c is the total number of labels.

Similarly, the statistic of some indicators based on samples can be obtained. For example, the multi-label accuracy rate Acc_{ML}, Hamming loss HmLoss, 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

formula is 2.15-2.19.

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

$$\mathrm{HmLoss} = \frac{1}{n}\sum_{i=1}^{n}\frac{\mathrm{XOR}(Y_i, \tilde{Y}_i)}{c} \tag{2.16}$$

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

$$\mathrm{Recall}_{\mathrm{ML}} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap \tilde{Y}_i|}{|Y_i|} \tag{2.18}$$

$$\mathrm{F1}_{ML} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{2.19}$$

2.3.4 Setting and Environment

The distribution of the number of exercise knowledge points in this data set is shown in the figure. The number of questions involved in different knowledge points is different. The set of questions involved in knowledge points j is defined as $\mathbf{E}_j$. Remember the threshold of the number of occurrences of the knowledge point label $\tau^{(KP)}$, then the knowledge points of the exercises can be divided according to the frequency of occurrence, i.e., $\{j|\mathbf{E}_j| > \tau^{(KP)}\}$ and $\{j||\mathbf{E}_j| \leq \tau^{(KP)}\}$. According to the different thresholds, the model's ability to classify knowledge points frequently and appear sparse can be tested separately.

In the experiment, five sets of thresholds of 200, 150, 100, 50, 10 are set $\tau^{(KP)}$, and the exercises with the label j are recorded as $\mathbf{E}_j$, the label set that meets the requirements can be counted as $\mathbf{L}_\tau^{(KP)} = \{j \mid |\mathbf{E}_j| \geq \tau^{(KP)}\}$, the exercises containing label in $\mathbf{L}_\tau^{(KP)}$ is denoted as $\mathbf{E}_\tau^{(KP)}$, within which the average number of knowledge points of the exercise in the set is $\overline{L}$.

Table 2.1 Setting of experiment

$\tau^{(KP)}$	$\lvert\mathbf{L}_{\tau^{(KP)}}\rvert$	$\lvert\mathbf{E}_{\tau^{(KP)}}\rvert$	$\overline{L}$
200	2	463	1.21
100	22	1376	1.55
50	29	2237	1.42
10	57	3158	1.35

The running environment are shown in Table 2.2.

Table 2.2 Experiment running environment

Software/Hardware	Configuration
CPU	i7 9700K
GPU	Tesla V100
Operating System	Ubuntu 20.04
Python	3.8.6
PyTorch	1.6.0
GPU Driver	Cuda10.1/cudnn7

There are many adjustable parameters of the model in BERT and GCN. Hyperparameters are shown in Table 2.3.

Table 2.3 Hyperparameter settings of recommendation model

Hyperparameter	Value
Transformer Layers	12
Dimension of hidden layer	768
Optimizer	Adam
Learning Rate	0.001
LSTM dimension	150
batch size	32
Dropout	0.3

2.3.5 Result and Analysis

The comparison experiment results are divided into the performance comparison of the horizontal baseline model (obtained in Table 2.4-2.7).

Table 2.4 Result comparison ($\tau^{(KP)} = 200$)

Metrics	$F1_{macro}$	$F1_{micro}$	Acc_{ML}	HmLoss	$F1_{ML}$
NB	75.3	74.2	69.6	18.2	73.6
ML-KNN	77.1	76.2	73.2	17.4	76.3
CNN+word2vec	79.5	78.4	76.6	14.2	79.6
CNN+BERT	80.1	**79.9**	76.9	13.7	79.5
Proposed	**80.9**	79.1	**77.3**	**13.1**	**80.7**

It can be seen from the table that the performance of the model proposed in this thesis is stronger than the baseline model on the exercise training machine with a higher frequency.

Table 2.5 Result comparison ($\tau^{(KP)} = 100$)

Metrics	$F1_{macro}$	$F1_{micro}$	Acc_{ML}	HmLoss	$F1_{ML}$
NB	71.2	72.1	67.2	16.2	71.8
ML-KNN	73.2	72.3	69.1	15.9	74.7
CNN+word2vec	74.3	74.4	72.3	13.2	75.2
CNN+BERT	74.4	74.6	72.3	13.1	75.1
Proposed	**75.5**	**75.7**	**73.1**	**12.7**	**74.9**

Table 2.6 Result comparison ($\tau^{(KP)} = 50$)

Metrics	$F1_{macro}$	$F1_{micro}$	Acc_{ML}	HmLoss	$F1_{ML}$
NB	52.3	53.0	42.1	9.2	51.9
ML-KNN	44.2	43.9	23.5	10.1	42.1
CNN+word2vec	56.1	**57.3**	46.2	8.2	56.5
CNN+BERT	56.2	56.8	**47.0**	**8.1**	56.1
Proposed	**57.1**	57.2	45.2	8.6	**57.5**

Table 2.7 Result comparison ($\tau^{(KP)} = 10$)

Metrics	$F1_{macro}$	$F1_{micro}$	Acc_{ML}	HmLoss	$F1_{ML}$
NB	36.5	37.1	26.1	4.2	36.5
ML-KNN	30.1	32.1	29.1	3.6	32.5
CNN+word2vec	36.7	38.2	37.5	3.5	37.2
CNN+BERT	36.9	**38.6**	**38.6**	**3.5**	37.5
Proposed	**37.1**	38.3	35.4	3.8	**38.6**

As the frequency of knowledge points decreases, classification difficulty increases and performance degradation occurs in all models. The reason is that the model cannot effectively capture information for fewer tags, resulting in increased errors. In general, the model proposed in this thesis has achieved the best or better performance in terms of F1-Score parameters and stricter subset accuracy indicators. When the problem labels' frequency is shallow, almost all models cannot achieve good results. This is because due to the uneven frequency distribution of the problem labels, a small number of unpopular labels cannot be well labeled, resulting in over-fitting in model training. This phenomenon affects labeling performance. In order to solve this problem, the model prediction performance can be optimized by using a larger set of exercises with a more average frequency of knowledge points than the training set.

2.4 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 the manual labeling cost is high and the efficiency is low. Therefore, automatic marking of knowledge points of test questions has become an urgent 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. A better knowledge point labeling effect 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 represented by a graph neural network, which can dig out the hidden knowledge point labels in the original text and provide the existing model with the knowledge point label association reasoning function.

2. By designing the correlation function between the knowledge points, the dependence between the knowledge points is modeled, and good performance has been achieved.

3. It is verified that the low-knowledge point label frequency will produce an over-fitting phenomenon for the multi-label classification model, resulting in performance degradation. It can be solved by averaging the label frequency of the training data set.

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 tracking model to track students' knowledge state and can also be used as the recommendation system's input feature to complete the recommendation of 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. Knowledge tracing is used to model students' knowledge mastery based on their previous answer records to obtain their knowledge status. There is a wealth of models for knowledge tracing, early models of knowledge tracing Bayesian Knowledge Tracing (BKT) [9], 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) [37] 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) [38] 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. However, these models still suffer from the problem of long dependencies. Sequential Key-Value Memory Networks (SKVMN) [39], which designs a hop-LSTM structure to aggregate the hidden states of similar exercises, was proposed. However, existing models often do not consider enough the higher-order connections between knowledge points, or treat knowledge points as mutually independent nodes, or treat knowledge points as simple hierarchical models. However, knowledge points are higher-order graph-like structures, in which case, using graph neural networks to characterize the relationships between knowledge points, training problem embedding and concept embedding is a better choice, so in this chapter, a graph neural network-based knowledge tracing model, which uses higher-order problem-knowledge point relationships to solve the sparsity and complex knowledge point dependency problems, and at the same time, it can learn the mastery speed of students through the problem logging module, to better characterize the knowledge tracing process.

3.2 Related Theory

3.2.1 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 $X_t = (x_1, x_2, \ldots, x_{t-1})$ 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 $P(a_t = 1|a_{t-1}, \ldots, a_1)$. As shown in the figure Fig.3.1.

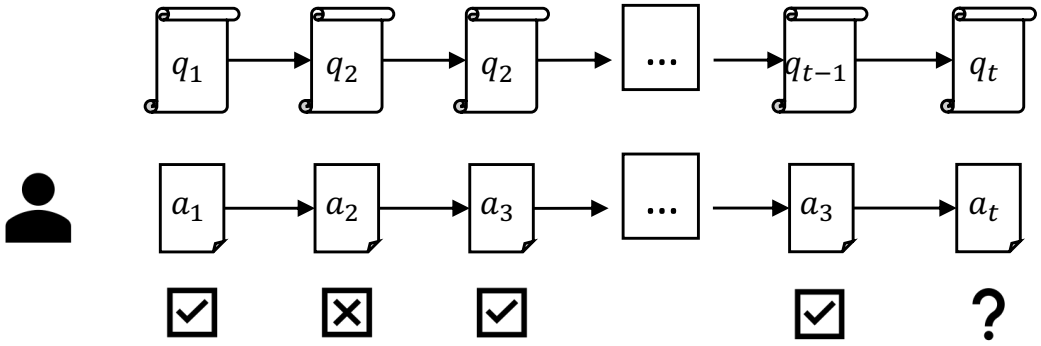


Fig. 3.1 The knowledge tracing pattern

3.2.2 Graph Neural Networks

The traditional neural network model, which has achieved tremendous success in extracting Euclidean space data features, has not worked well for non-Euclidean space applications. Moreover, currently, many practical application scenarios are formally generated in non-Euclidean spaces. In the process of knowledge tracing, the connection between knowledge points is exactly a structure of non-Euclidean space, i.e., knowledge graph structure. The irregularity and complexity of graphs pose a significant challenge to traditional deep learning algorithms, and the variability of the connections of the nodes of graphs leads to a massive computational effort, which makes some traditional neural network models that perform well in Euclidean space cannot be directly applied to non-Euclidean space. e.g.,

in processing images where each pixel of the image is independent of each other, convolutional neural networks are easier to compute and parallelize for structures like images that are stable and single and have individual parts independent of each other, but this is not the case for graph structures where each vertex is connected to and has dependencies on other nodes. Graph-based Networks are proposed to characterize the interdependencies between graph nodes.

As the pioneer of graph neural networks, GCN applies the convolution operation in image processing to graph-structured data processing and gives a specific derivation. It aggregates the features of graph neighbor nodes to make a linear transformation. Multiple GCN layers can be stacked to capture the k-hop neighbor vertex information to solve the problem of insufficient graph propagation depth. However, GCN processing requires putting the whole graph into the computational memory, limiting its computational performance. It also requires pre-inputting the graph structure information, which limits its application. To solve the problem, GCN aggregates neighbor nodes ignoring different neighbor nodes' weights, Graph Attention Networks (GAT) attention algorithm to compute each vertex's representation. It does not need to use a pre-constructed graph, so only the neighbor nodes need to be known, significantly increasing the computational speed.

Graph embedding is an important research topic in graph neural networks, representing nodes in a graph as low-dimensional vectors by preserving the network topology and vertex information of the graph, which is then processed by other machine learning algorithms. In this chapter, the association between knowledge points and knowledge point information by a graph embedding algorithm to obtain the learning of embedding for knowledge structures is characterized.

3.3 Proposed Model

3.3.1 Algorithm Overview

The key point of this section is to track students' knowledge. What is issued is a knowledge tracing model based on graph attention network and Transformer model. In the experimental verification phase, the performance of the model on several datasets achieves the best performance. The first layer of this model is the embedding aggregation layer. In this layer, a GAT embedding is applied to aggregate the problem embedding and the knowledge point embedding and how the internal relationship between the problem and the knowledge point. It can solve the problem of knowledge point dependence and complex relationships. The second layer is a transformer-based knowledge tracing model. Referring to the famous

transformer model proposed by Google Brain [40], this layer designs a knowledge tracing model with Transformer-structure, which inputs exercises and skills embedding and outputs the prediction correctness to the next exercise. Through this mechanism, the model can realize the learning of long-range dependence. Also, the model can output the hidden knowledge state through the Decoder, which is the Key to exercise recommendation in the next stage. In a word, the overall architecture diagram of the model is in Fig.3.3, which is segmented into the following modules:

1. GAT-based embedding layer: embedding layer based on gat: this layer uses the embedding method to represent exercises, knowledge points, and answers. In the graph neural network, each problem represents a vertex connected with several knowledge nodes, and these knowledge nodes are connected with the related problem nodes. Due to the relevance of knowledge points, they can be directly connected with other ones. The structure is shown in the Fig.3.2. This embedding layer does not carry out pretraining but carries out overall training with other layers to optimize the final result.

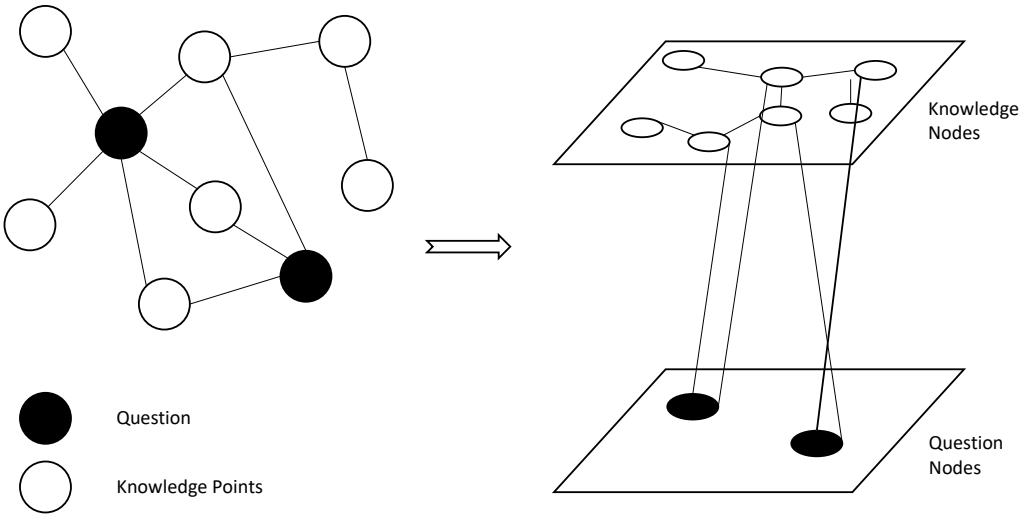


Fig. 3.2 The relation graph between knowledge point and exercise

2. Transformer-based knowledge state encoder: this layer establishes a graph of knowledge points applied to track the students' mastery proficiency to knowledge points. This layer updates the knowledge state through the knowledge state, and finally, it will be applied in the subsequent recommendation process.

3. Knowledge tracing layer based on Transformer: this layer is similar to the traditional transformer structure and has two parts: encoder and Decoder. It learns the weight matrix of knowledge points and problems and can solve knowledge point dependence. Through the attention mechanism, it can also capture similar problems of knowledge points.

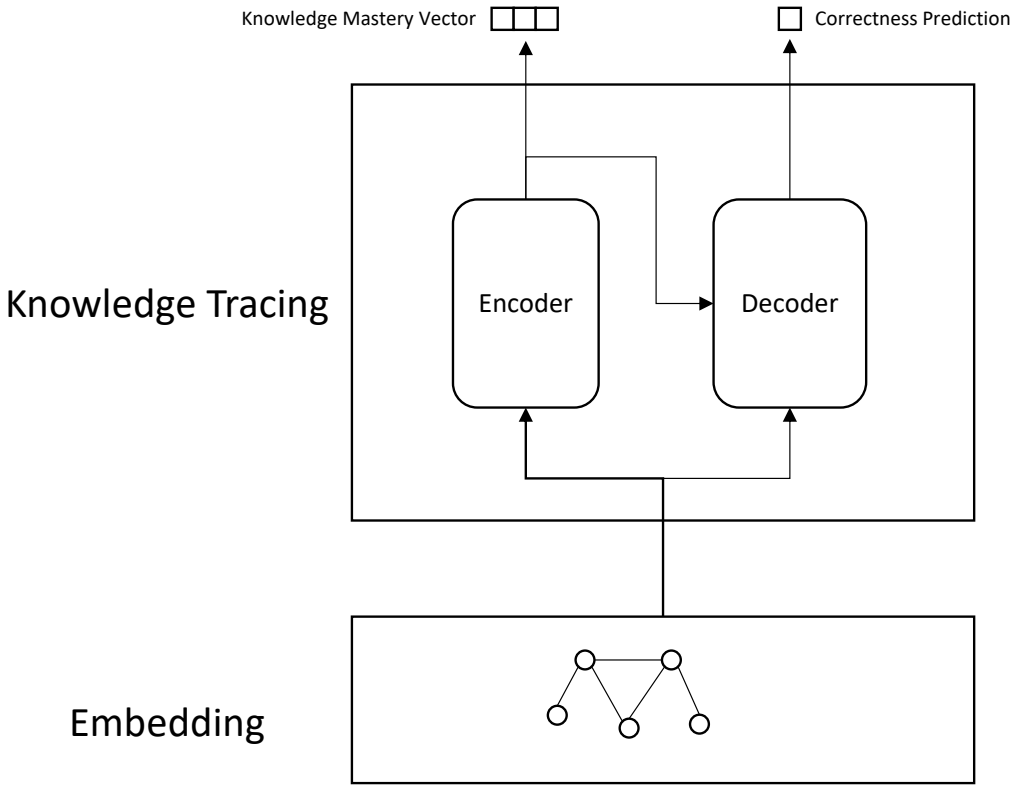


Fig. 3.3 Model structure overview

3.3.2 GAT-based Embedding Layer

In this section, a graph self-attention network (GAT) is introduced to embed the problem-knowledge point association graph, reducing the data sparsity of the knowledge points in the problem data. The ith exercise can be recorded as q_i, the student's answer to the ith exercise as a_i, and the answer is correct or wrong, so it can be recorded as $a_i \in \{0, 1\}$, the answer record is recorded as $X = \{x_1, x_2, \ldots, x_{t-1}\}$, where $x_i = (q_i, a_i)$. It is based on the previous $t-1$ times to predict the exercise's correctness in t. For each exercise q_i, there are a series of knowledge points related to it. These knowledge points are denoted as $\{p_1, p_2, \ldots, p_{n_i}\}$. The

embedding calculation can obtain the deep-level relationship between the problems, i.e., the relevance of knowledge points. This can be done ideally with a graph structure. Intuitively, the probability of answering the exercise can be inferred through mastery proficiency to the knowledge points. This method can also establish a better representation of the problem-knowledge point.

Embedding calculation can obtain the deep correlation between problems, i.e., the correlation of referred knowledge points. This can be achieved with graph structure. Intuitively speaking, it also infers the correct answer probability of the exercise by mastering the knowledge points. This method can also establish a better representation of problem knowledge points. In the exercise-knowledge graph, GAT can capture the high order relation for n-hop neighboring nodes. Also, GAT uses the idea of Transformer for reference and introduces a masked self-attention mechanism. When calculating each vertex's representation in the graph, it will assign different weights to its neighbors according to their different characteristics.

GAT applies an attention mechanism to the weighted summation of neighboring vertex features. The weights of the neighboring vertex features depend entirely on the vertex features and are independent of the graph structure. In GAT, each vertex in the graph can be assigned different weights to neighboring nodes based on their characteristics. With the introduction of the attention mechanism, it is only relevant to adjacent nodes, i.e., nodes that share edges, without getting information about the whole graph. Specifically, the vertex of the GAT network represents the knowledge point embedding or exercise embedding.

The proposed method treats all the exercises as nodes of GAT. For the exercises i in an exercise set of size $N^{(e)}$, the input features are $\vec{h}_i^{(e)}$, and the features include the location of the exercises, the correctness of the answers to the exercises, and the knowledge points contained in the exercises. The adjacency matrix A of the exercises can be established from the exercises category, i.e., when multiple exercises are of the same category, they are neighbor nodes, so preprocessing can reduce the subsequent computation and prevent the interference caused by invalid connections. For example, when both Exercise 1 and Exercise 3 belong to the category of linear algebra, $A[1,3] = 1$, they will be aggregated in the graph computation for features. The neighboring vertex set of vertex i is marked as $\mathscr{V}^i$. Input of GAT can be denoted as $\mathbf{h}_0^{(e)} = \{\vec{h}_{0,1}^{(e)}, \vec{h}_{0,2}^{(e)} \ldots, \vec{h}_{0,N^{(e)}}^{(e)}\}$, in which $\vec{h}_i^{(e)} \in \mathbb{R}^{F^{(e)}}$ and $F^{(e)}$ is the amount of feature of exercise.

To achieve better high dimension feature representation, at least one linear transformation from low-dimensional should be embedded to high-dimensional features. Therefore, a linear transformation is first done for the vertex features and then the coefficients are calculated. The attention mechanism of self-attention is implemented for each vertex, and the atten-

tion coefficients (Attention coefficients) and its softmax value are calculated in formula 3.1 and 3.2:

$$e_{ij} = a(\mathbf{W}_l \vec{h}_i^{(e)}, \mathbf{W}_l \vec{h}_j^{(e)}) \tag{3.1}$$

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \tag{3.2}$$

The e_{ij} is the contribution degree from vertex i to vertex j and the α_{ij} is the normalized contribution degree, and the $\mathbf{W}_l$ is the ith layer's weight matrix. Here, since the graph structure information is to be considered, the concept of masked attention is introduced, which means that the denominator of the softmax considers only the first-order information about the neighbors. By this method, each vertex then gets a weight about the neighboring nodes, and then the combined representation of the nodes is obtained by summing. That is, the attention weighted summing can be calculated to generate aggregate neighboring features. The relative formula is 3.3, where σ are the non-linear transformation such as ReLU or LeakyReLU. If the edge $j \to i$ does not exist, the calculation of α_{ij}, which can reduce the calculation, can be performed. The formula means that this vertex's output characteristics are related to all the nodes adjacent to it and are obtained after the nonlinear activation of their linear sum.

$$\vec{h}_{l+1,i}^{(e)} = \sigma(\sum_{j \in \mathcal{N}^i} \alpha_{ij} \mathbf{W}_l \vec{h}_{i,j}^{(e)}) \tag{3.3}$$

To make the result of self-attention more stable. The multi-headed attention mechanism can be used here.K independent attention mechanisms can be achieved by connecting k features together. The formula is like 3.4.

$$\vec{h}_{l+1,i}^{(e)} = \|_{k=1}^{K} \sigma(\sum_{j \in \mathcal{N}^i} \alpha_{ij}^k \mathbf{W}_l^k \vec{h}_{i,j}^{(e)}) \tag{3.4}$$

K-averaging is used to replace the join operation and to delay the application of the final nonlinear function. The attention coefficients between different nodes after regularization are obtained by the above operation and can predict each vertex's characteristics. The formula is like 3.4.

$$\vec{h}_{l+1,i}^{(e)} = \sigma(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}^i}\alpha_{ij}^k \mathbf{W}_l^k \vec{h}_{i,j}^{(e)}) \tag{3.5}$$

From L-layer GAT, the embedded vector of exercise i are denoted as $E_i^{(e)} = \vec{h}_{L,i}^{(e)}$, which would be passed into the Transformer Block in Knowledge tracing module. The general process of embedding layer is in Fig.3.4

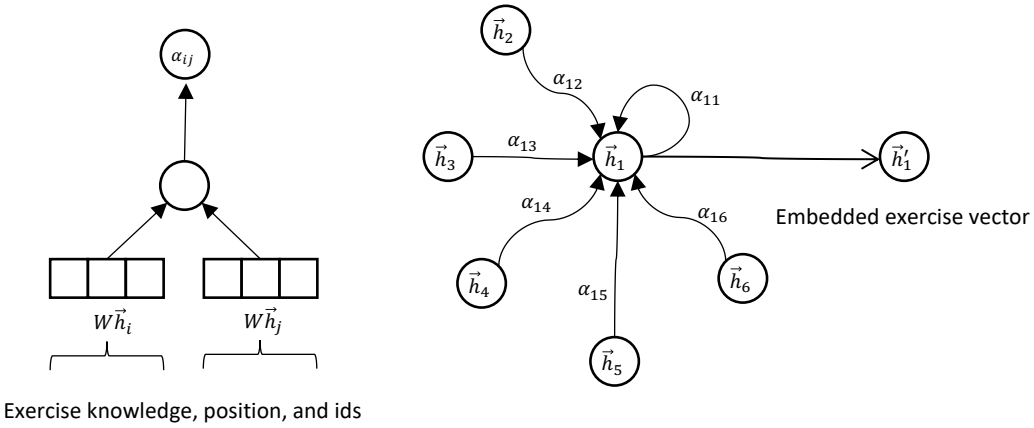


Fig. 3.4 General process of embedding layer

3.3.3 Transformer Based Knowledge Tracing

The input of transformer Block is the concatenation of the embedded exercise $E_i^{(e)}$ and corresponding answer correctness a_i, i.e. $E_i = E_i^{(e)} \| a_i$. The general structure is like Fig.3.5. The traditional Transformer model is modified to apply to the knowledge tracing task. The Transformer can learn the W-matrix relating underlying knowledge points and exercises. This allows the model to learn deeper connections between problems, resulting in better inference performance. It can cluster problems with similar knowledge points and reduce the variance for less frequent problems.

The output of embedding layer E_i is passed into a Transformer Block, of which the first is a masked multi-head attention (MHA) module. The formula is 3.6.

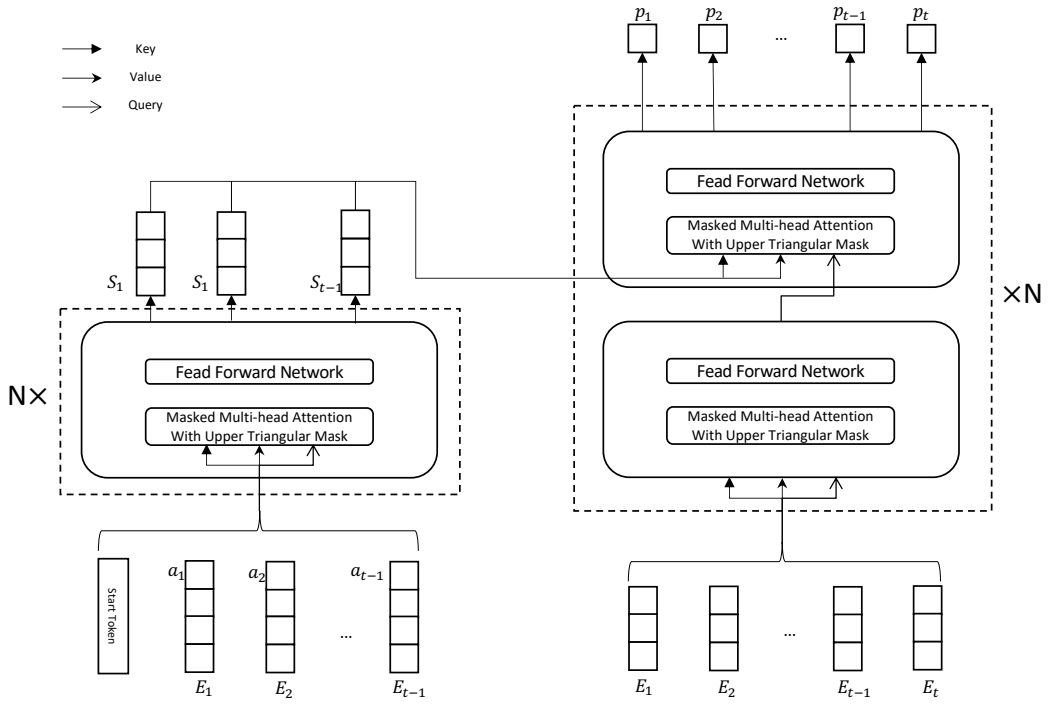


Fig. 3.5 The knowledge tracing architecture

$$
\begin{aligned}
q_i &= W_i^{(Q)} E_i \\
k_i &= W_i^{(K)} E_i \\
v_i &= W_i^{(V)} E_i \\
A_{ij} &= \frac{q_i k_j + b(\Delta t_{i-j})}{\sqrt{d_k}}, \forall j \leq i \\
h_i &= \sum_{j \leq i} \text{softmax}(A_{ij}) v_j
\end{aligned}
\tag{3.6}
$$

Query, Key, and Value sequences are fed into the model. It separately extracts value v_i, query q_i, and key k_i. Then the attention from i to j can be calculated. The time difference $b(\Delta t_{i-j})$ imports time variance into attention of i and j. Then the head i representation, which is h_i, can be obtained. The upper masking should be added into the MHA model to avoid the effect from the current state to the future state.

The MHA output is formula 3.7, where $N^{(\text{MHA})}$ is the number of head and $W^{(Aggregate)}$ is the aggregate weight matrix. The $\|$ represents concatenation.

$$\text{MHA}(Q, K, V) = (h_1 \| h_2 \| \dots \| h_t) \times W^{(Aggregate)} \tag{3.7}$$

Then it is needed into the feed-forward net (FFN), which is for non-linearity import. The formula is 3.8, where the knowledge state vectors $S_i = F_i$.

$$F = \text{FFN}(\text{MHA}(Q, K, V)) \tag{3.8}$$
$$= (F_1, F_2, \dots, F_t) \tag{3.9}$$

The output of FFN is actually the hidden knowledge mastery proficiency state vector. It is the output of the encoder module. In general, the Encoder formula is 3.10. The SkipCon$(\cdot)$ is the skip connection [41] which can reduces gradient disappearance and network degradation problems, making training easier. The LNorm$(\cdot)$ is the Layer normalization [42] for normalization.

$$\begin{aligned} M^{(En)} &= \text{SkipCon}(\text{MHA}(\text{LNorm}(Q, K, V))) \\ O^{(En)} &= \text{SkipCon}(\text{FFN}(\text{LNorm}(M))) \end{aligned} \tag{3.10}$$

The $O^{(En)}$ is then as the input of Decoder's Key and Value. The Decoder apply one MHA to generate a representation of question information as formula 3.11

$$\begin{aligned} M_1^{(De)} &= \text{SkipCon}(\text{MHA}(\text{LNorm}(Q', K', V'))) \\ M_2^{(De)} &= \text{SkipCon}(\text{MHA}(\text{LNorm}(M_1^{(De)}, O^{(En)}, O^{(En)}))) \\ O^{(De)} &= \text{SkipCon}(\text{FFN}(\text{LNorm}(M_2^{(De)} M))) \end{aligned} \tag{3.11}$$

The $M_i^{(De)}$ is the ith MHA block output of decoder module. The output of decoder $O^{(De)}$ is the prediction for exercise answering. The prediction is $\mathbf{p} = (p_1, p_2, \dots, p_t)$

3.4 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 met-

rics, experimental settings, and operating parameter environments for model performance comparison, and finally draws the results and analyzes them.

3.4.1 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. Dataset statistics are shown in Table 3.1.

- ASSISTment 2015 (ASSIST15) is a data set used to predict student test performance. It consists of a series of exercises containing several knowledge point labels and some student answer records. It was collected on an online education platform in 2015. After data filtering, a data set containing 19,917 students, about 102 thousand exercises, 100 knowledge point labels, and about 709 interactions was obtained.
- ASSISTment 2017 (ASSIST17) is derived from the ASSISTments Longitudinal Data Mining Competition held in 2017. After data preprocessing and data filtering, the data set contains 1709 students, 4117 exercises, 102 knowledge points, and approximately 943 thousand interactions.
- Statics 2011 (STATICS11) dataset is derived from the answer records of statistics courses. It contains 333 students, 1223 exercises, 156 knowledge tags, and about 189 thousand interactions.

Table 3.1 Dataset statistics

Dataset	#Students	#exercises	#Knowledge Points	#Interactions
ASSIST15	19,917	102,396	100	709K
ASSIST17	1,709	4,117	102	943K
STATICS11	333	1,223	156	189K

3.4.2 Settings and Metrics

At present, most papers in the field of knowledge tracing use Area Under Curve (AUC) as a comparison indicator. AUC is a commonly used two-category evaluation method based on the ROC curve and has good performance comparison capabilities. In the task of knowledge

tracing, the final prediction is also 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.

For each dataset, 70% is used as the training set, and the remaining 30% is taken as the test set. The experimental running environment is Ubuntu20.04, TensorFlow 2.23, Python 3.8.6, and the experimental machine is a Tesla V100 computing card.

3.4.3 Baselines

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

- BKT [9]: BKT is a probabilistic model that predicts the probability of correctness of answers according to the basic assumption that knowledge points are independent.
- DKT [10]: 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) [38]: 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.
- Graph-Based Knowledge Tracing (GKT) [11]: GKT applies graphical neural networks to knowledge tracing tasks, using the properties of graphs to characterize the graph-like associations between knowledge points for better learning of intrinsic knowledge dependencies, and the model is able to learn students' hidden knowledge states and also has better interpretability.

3.4.4 Result and Analysis

The experiment's result is shown in Table 3.2, and the proposed model achieves better performance metrics than other baseline models. The labels of DKT, DKVMN, and GKT are significantly better than the BKT model on all datasets. On the ASSIST15 dataset, the AUC value of the proposed model is 0.5% higher than the other models. At the same time, GKT outperforms DKVMN and DKT due to the high hidden association of knowledge

points in this dataset, so the model that can better characterize the intrinsic association of knowledge points significantly produces better results. On the ASSIST17 dataset, the proposed model slightly outperforms the compared models. Simultaneously, the performance difference between DKT, DKVMN, and GKT is minor due to the weak association between knowledge points, and thus the performance improvement from considering the association between knowledge points is more diminutive. On the STATICS11 dataset, the performance of DKVMN outperforms GKT due to the overfitting phenomenon arising from modeling the relationship to knowledge, thus negatively affecting the performance, and the results show that the proposed model has better performance against overfitting.

Table 3.2 AUC results (%) over three datasets

Model	ASSIST15	ASSIST17	STATICS11
BKT	62.01 ± 0.03	65.30 ± 0.01	64.21 ± 0.01
DKT	70.83 ± 0.03	72.66 ± 0.01	72.46 ± 0.06
DKVMN	71.06 ± 0.03	72.78 ± 0.02	72.67 ± 0.03
GKT	72.12 ± 0.02	72.85 ± 0.01	72.57 ± 0.01
Proposed	$\mathbf{72.62 \pm 0.02}$	$\mathbf{72.89 \pm 0.02}$	$\mathbf{72.73 \pm 0.02}$

3.5 Summary

In this section, a knowledge tracing model based on graph self-attentive network hungry embedding with Transformer is proposed, which learns the relationship between exercises and knowledge points through GAT, which can effectively characterize the complex dependencies between knowledge points and can discover the hidden knowledge points in the exercises. In the encoding part, the embedding vectors learned by GAT are encoded by a Transformer model for knowledge state, and a knowledge state sequence is an output, which can be used in the subsequent exercise recommendation process. Also, the knowledge state sequence and the exercises are used as the subsequent decoder's input to output a simulation of the correct or incorrect answers to the exercises.

The innovation points of this section are as follows.

1. Based on the graph self-attentive network, a relational graph network with separation of exercises and knowledge points are designed, and the knowledge state embeddings learned in this way can better characterize the intrinsic dependencies of knowledge points.

2. Through the encoder based on Transformer block, the intrinsic knowledge state vector of students is output, and this state vector can be used for subsequent learning resource recommendation.

3. With the Transformer block, bidirectional sequence information can be learned better to model students' knowledge states' change.

The model is experimentally verified to achieve good prediction performance for both datasets with closely linked knowledge points and those with sparse knowledge point relationships, and overfitting can also be prevented by proper tuning of parameters.

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 the user's demographic characteristics, and 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 greatly 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, thereby 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 is decreasing. 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 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, etc., 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 [43], factorization machines (FM) [44] and weighted matrix factorization(WMF) [45], 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 [46], DeepCross [47], DeepFM [48], etc. 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.

4.2.1 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 prediction of the exercise answer, and generate a final exercise recommendation list.

The structure of the recommendation system is like Fig.4.1

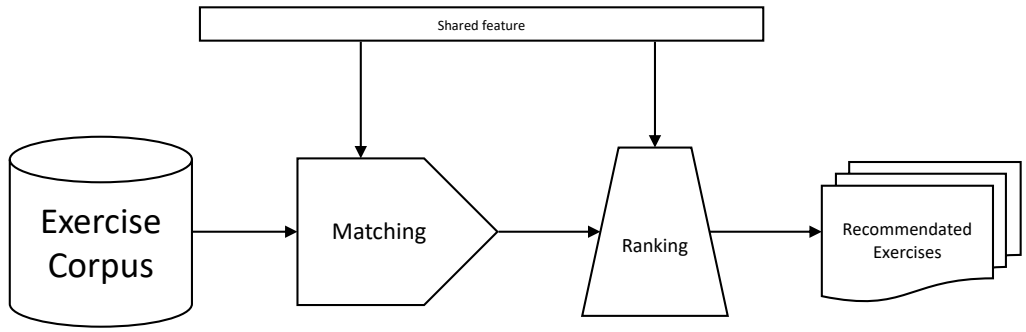


Fig. 4.1 The architecture of recommendation system

4.2.2 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 stu-

dents' 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 definitely 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 submatching exercise set, the exercises are merged and filtered through the weighted merging algorithm. Its structure is shown in Fig.4.2.

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.

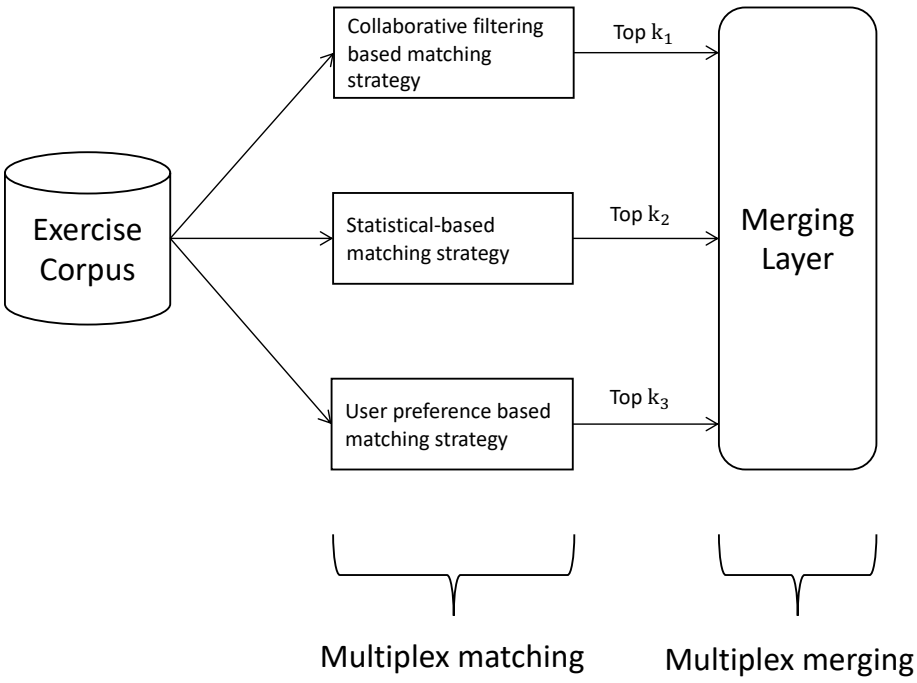


Fig. 4.2 The multiplex matching method

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 $\mathbf{S}^u = (S_1^u, S_2^u, \ldots, S_n^u)$, $\mathbf{S}^v = (S_1^v, S_2^v, \ldots, S_n^v)$ represent the rep-

resentation vectors of the knowledge state of users u and v output by the knowledge tracing model, respectively.

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

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 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, $\mathbf{S} = \{\mathbf{S}^1, \mathbf{S}^2, \ldots, \mathbf{S}^N\}$;
The mistaken/favorite exercise sets of all students $\mathbf{E} = \{\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_N\}$
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, in order 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 itemsets 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 recommendation scores of all sub-matching strategies. 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 comprehensive 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.

4.2.3 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.

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. 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.

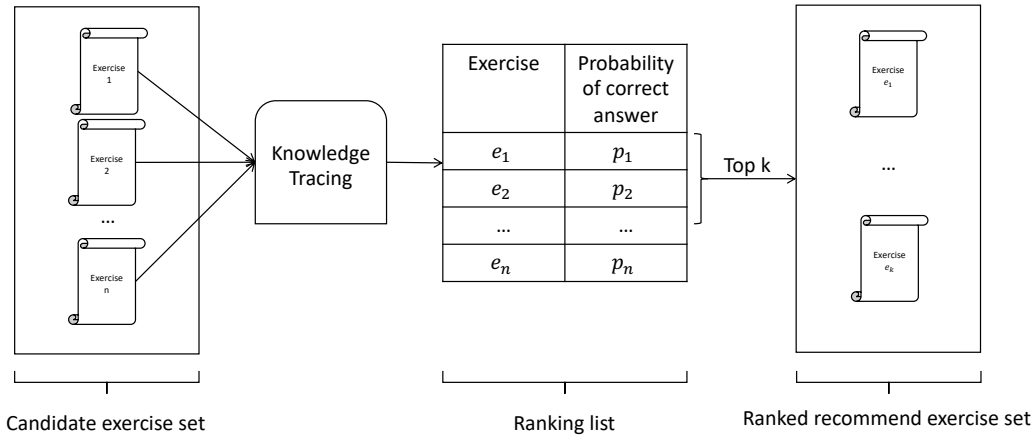


Fig. 4.3 The ranking method

The general algorithm is as 2:

Algorithm 2 Recommendation ranking algorithm

Require:
The candidate exercises set $\mathbf{E}^{(Candidate)} = \{E_1, E_2, \ldots, E_N\}$;
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 $\mathbf{E}^{(UCF)}$;

4.3 Experiments

This model's core idea is to enhance students' proficiency in the parts with weak knowledge mastery. The model should recommend to students the exercises that bring the greatest positive benefits to their knowledge mastery. A recommended dataset based on the public knowledge tracing dataset is generated for performance evaluation in the experimental phase.

4.3.1 Dataset

Considering that this recommendation system needs to recommend exercises based on students' knowledge status, there is currently no dedicated dataset for mathematics exercise recommendations based on knowledge mastery proficiency. This model generates recommendation data sets based on the knowledge tracing dataset. This experiment uses the unique

dataset ASSISTment2009 in the field of knowledge tracing. It is a dataset used to predict student test performance. It consists of a series of questions containing several knowledge point labels and some students' answer records. It was collected on the online education platform in 2009-2010. After data filtering, a dataset containing 4400 students, 110 knowledge point labels, and about 328,291 interaction times was obtained.

4.3.2 Experiment Setting

The original data set is a fixed sequence of question-answering interaction. To measure the recommendation performance, the exercise q_t answered in timestamp t can act as the calibration item to check whether it is in the final recommendation result under the simultaneous state. The final recommendation output obtained by the ranking model in the previous section is a list with a capacity of K, and the recommendation effect of the model can be determined by determining whether the exercises that predict the user will answer wrongly are in the list.

The running environment of this article is shown in 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. The hyperparameter settings of the recommendation model are shown in Table 4.2. In the ranking stage, parameter N represents the number of Transformer stacking blocks in the knowledge tracing model, and d represents the sub-layer output's dimension.

Table 4.1 Experiment running environment

Software/Hardware	Configuration
CPU	i7 9700K
GPU	Tesla V100
Operating System	Ubuntu 20.04
Python	3.8.6
PyTorch	1.6.0
GPU Driver	Cuda10.1/cudnn7

Baseline

As a comparison, the more popular Collaborative Filtering (CF) recommendation algorithm and the random recommendation algorithm are used as Baseline models.

- Collaborative Filtering: The collaborative filtering algorithm omits the matching stage and directly finds similar users' exercises as the recommendation set.

Table 4.2 Hyperparameter settings of recommendation model

Parameter Belonging	Hyperparameter	Value
Collaborative filtering matching strategy	$\tau^{simuser}$	0.9
	$K^{(UCF)}$	10
Popularity based matching strategy	$w^{(SA)}$	0.5
	$w^{(SF)}$	0.1
	$w^{(SW)}$	0.3
	$w^{(EX)}$	0.1
	K^{PMS}	10
User preference based matching strategy	$w^{(UW)}$	0.9
	$w^{(UF)}$	0.1
	K^{UPM}	10
Multiplex Merging	$w^{(UCF)}$	0.9
	$w^{(PMS)}$	0.05
	$w^{(UPM)}$	0.05
	$K^{(MERGE)}$	10
Ranking	$K^{(RANK)}$	10
	N	3
	d	256
Training	batch size	128
	epochs	50
	dropout rate	0.05
	window size	100

- Random: In each interaction, randomly determine whether the exercise will be recommended

4.3.3 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 accuracy of the recommendation is a key metric. In order to show more intuitively the knowledge enhancement effect of the recommendation algorithm on users, this experiment will be based on two aspects for model performance determination. Train the model by using the interaction sequence of the knowledge tracing dataset as the input of the knowledge tracing model. Simultaneously, the recommendation model is launched to generate a collection of exercise recommendations. Record the recommended results of the exercise $e_{i,t}$ for which the student s_i will be in the sequence of time t. If the prediction probability $p_{i,t}$ of the knowledge-tracking model for the exercise $e_{i,t}$ is lower than a given judgment threshold τ, it means that the exercise $e_{i,t}$ is error-prone for the student s_i and thus should be recommended. The confusion matrix is:

- True Positive(TP): exercise $e_{i,t}$ is in output recommendation exercise set and $p_{i,t} < \tau$
- True Positive(FP): exercise $e_{i,t}$ is in output recommendation exercise set and $p_{i,t} \geq \tau$
- True Positive(FN): exercise $e_{i,t}$ is not in output recommendation exercise set and $p_{i,t} < \tau$
- True Positive(TN): exercise $e_{i,t}$ is not in output recommendation exercise set and $p_{i,t} \geq \tau$

Where τ is an adjustable hyperparameter, this is a classification problem, so the Receiver Operator Characteristic Curve (ROC) curve can be used to determine the performance of the recommendation system. Then, (Area Under Curve) AUC and Accuracy (ACC) metrics can be used to determine the performance of the model.

4.3.4 Result and Analysis

Five different τ (0.45, 0.5, 0.55) are applied as the threshold for judging the predicting correctness of the exercises. The result is shown in Table 4.3. After comparison, it can be concluded that the knowledge tracing model-based exercise recommendation significantly outperforms the Baseline algorithm in both ACC and AUC metrics, which indicates that

Table 4.3 Recommendation experiment result

Model	ACC	AUC
CF	0.5375	0.5239
Random	0.5073	0.5012
Proposed ($\tau = 0.45$)	0.6542	0.6878
Proposed ($\tau = 0.50$)	0.6839	0.7375
Proposed ($\tau = 0.55$)	0.6657	0.7039

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.

4.4 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 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, 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

In this article, an exercise recommendation system based on knowledge tracing is designed. The system is divided into three modules: the exercise knowledge point tagging module, knowledge tracking module, and exercise recommendation module. The exercise knowledge point tagging module can mark the exercises' knowledge points without the knowledge points and use the marked knowledge points as the input data for knowledge tracking. The knowledge tracking 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 make targeted exercise recommendations.

In the knowledge point annotation module, a graph convolutional neural network is used to train the label annotation classifier. Use Bi-LSTM to mine the exercise text information, embed the exercise text as the label classifier group's input, and output a set of label classification results. Experiments show that the model has a good classification effect for high school math learning problems and has a specific improvement in existing models' performance. In future work, a Transformer can be applied to replace the Bi-LSTM model or BERT to train exercise text embedding to achieve more accurate classification results.

In the knowledge tracking module, a graph attention network is used to learn exercises' knowledge relationships. The embedding vector is output through the Transformer-based Encoder model to output a knowledge state vector, and then the exercise embedding vector and the knowledge state vector are used as the Transformer-based Decoder. The input and output of the model predict the answer to the next exercise. After experimental verification, the model has good predictive performance for data sets with complex and straightforward knowledge relationships. Later, consider using different graph neural network models as an optional method to increase learning efficiency or use more exercise information to learn exercise knowledge embedding.

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 employs 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 tracking 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.

In future work, for the exercise knowledge point labeling model, the popular pre-training models such as BERT, GPT can be applied for embedding learning and named test mining, which can provide a deeper semantic understanding of the exercise text and achieve better results. For the knowledge tracking model, applying other graphical neural network models can be an option to embed the learning of exercise - knowledge point graphs to quantify the relationship of knowledge points to the model. An explicit knowledge mastery matrix can be designed to visualize students' mastery of specific knowledge points. For the exercise recommendation model, neural network models can adaptively adjust each matching strategy's hyperparameters to reduce manual tuning costs.

References

[1] 马相春, 钟绍春, and 徐妲. 大数据视角下个性化自适应学习系统支撑模型及实现机制研究. 中国电化教育, 4:97–102, 2017.

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

[3] 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.

[4] 杨东明, 杨大为, 顾航, 洪道诚, 高明, 王晔, et al. 面向初等数学的知识点关系提取研究. 华东师范大学学报*(*自然科学版*)*, 5:53–65, 2019.

[5] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[6] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[7] Susan E Embretson and Steven P Reise. *Item response theory*. Psychology Press, 2013.

[8] Jimmy De La Torre. Dina model and parameter estimation: A didactic. *Journal of educational and behavioral statistics*, 34(1):115–130, 2009.

[9] 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.

[10] 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.

[11] 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.

[12] Huiyi Tan, Junfei Guo, and Yong Li. E-learning recommendation system. In *2008 International Conference on Computer Science and Software Engineering*, volume 5, pages 430–433. IEEE, 2008.

[13] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

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

[15] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, 2006.

[16] Matthew R McLaughlin and Jonathan L Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–336, 2004.

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

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

[19] 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.

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

[21] 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.

[22] 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.

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

[24] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*, 2019.

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

[26] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005.

[27] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[28] 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.

[29] 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.

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

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[37] 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.

[38] 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.

[39] Ghodai Abdelrahman and Qing Wang. Knowledge tracing with sequential key-value memory networks. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul 2019.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[42] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction, 2017.