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 Engineering

Central China Normal University Wollongong Joint Institute

Central China Normal University

February 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
February 2021

Abstract

This paper implements a high school mathematics learning resource recommendation system based on knowledge tracking and factorization machine, which contains three core points. The first one is the design of data source, and proposes an automatic extraction method for test knowledge points. The whole process of high school mathematics knowledge mapping from corpus collection, construction, storage, and update is introduced as well as the automation process. The second point is the implementation of a GCN-based knowledge tracking algorithm, which takes into account the a priori structure of knowledge points as well as students' recent question records and knowledge mastery changes, and achieves a more excellent performance compared to other knowledge tracking models. The third point is based on a deep factorization machine, which solves the problem of sparsity of training data, and it takes the output of the knowledge tracking model as input and considers various other feature inputs to achieve learning resource recommendation.

Keywords: Learning Resource Recommendation System，Knowledge Point Tagging，Knowledge Tracing，Learning Path Planing

摘 要

中文摘要...

关键词：keyword1，keyword2，keyword3，keyword4

Acknowledgments

Table of content

Figures

Tables

Nomenclature

Subscripts

crit Critical state

Acronyms / Abbreviations

ALU Arithmetic Logic Unit

BEM Boundary Element Method

CD Contact Dynamics

CFD Computational Fluid Dynamics

CK Carman - Kozeny

DEM Discrete Element Method

DKT Draft Kiss Tumble

DNS Direct Numerical Simulation

EFG Element-Free Galerkin

FEM Finite Element Method

FLOP Floating Point Operations

FPU Floating Point Unit

FVM Finite Volume Method

GPU Graphics Processing Unit

LBM Lattice Boltzmann Method

LES Large Eddy Simulation

MPM Material Point Method

MRT Multi-Relaxation Time

PCI Peripheral Component Interconnect

PFEM Particle Finite Element Method

PIC Particle-in-cell

PPC Particles per cell

RVE Representative Elemental Volume

SH Savage Hutter

SM Streaming Multiprocessors

USF Update Stress First

USL Update Stress Last

Chapter 1

Introduction

1.1 Research Background and Significance

Artificial intelligence industry has developed rapidly in recent years, it is being commercialized in all aspects, triggering profound changes in various industries, and the future development of artificial intelligence will be the combination of key technologies and industries.[7] At present, AI technology has been implemented in many fields such as finance, medical and security, and the application scenarios are becoming more and more abundant. The commercialization of AI has played a positive role in accelerating the digitization of enterprises, improving the structure of the industrial chain, and increasing the efficiency of information utilization. The traditional education industry also tries to use AI technology to help the development of the industry. Every development of AI is accompanied by breakthroughs in research methods, and deep learning is one of the important representatives of the breakthroughs in machine learning technology in recent years. With the continuous extension of human AI research and application fields, AI will usher in more kinds of technology combination applications in the future. Artificial intelligence has also begun to be applied to the education industry, and the concept of intelligent education has emerged. Among the types of applications of AI technology in education, AI adaptive learning is the most widely used in all aspects of learning. In addition, due to China's large population base, the shortage of educational resources, the importance attached to education and other favorable factors intelligent adaptive learning system is expected to come later.

In recent years, domestic adaptive learning has begun to enter the minds of many people involved in education training and education investment. There are more and more education

technology companies in the market that focus on adaptive learning tools. At the same time, many education companies have started to use adaptive learning as the main core function or main selling point of their products. The biggest advantage of adaptive education is that it can locate the knowledge gaps of each student. The adaptive learning platform will guide the student to the next most suitable learning content and activities for him. When students encounter a course that is too difficult or too low in the learning process, they can automatically adjust the difficulty of the course. Teachers can also analyze the knowledge gaps of each student based on the learning status evaluation report provided by the system, adjust the learning progress in real time, and provide personalized teaching for each student. So theoretically, adaptive learning is one of the potentially feasible solutions to the problem of "teaching to students according to their abilities" in online education. To make a practical adaptive learning system, I plan to use knowledge tracing to track students' learning status and use the factorization machine algorithm to calculate the relevance of topics to students to build such a test recommendation system.The current personalized learning resource recommendation system is one way of implementing adaptive learning, which is the subject of this paper.

In this paper, the study focuses on the recommendation of learning resources for the subject of high school mathematics. In this system, there are two aspects in general: on the one hand, scientific and targeted acquisition and tracing of students' knowledge state, and on the other hand, recommendation of personalized learning resources based on students' knowledge mastery state. We use the knowledge tracing algorithm of graph neural network to acquire and track students' knowledge states, and the factorization agent to try to combine the output of graph neural network with prior knowledge for resource recommendation.

1.2 Research Status

The dominant content of the research is knowledge tracing and recommendation system. Some advanced graph neural network algorithm is applied to finish the task. There have been some research advances and related applications in the area of knowledge tracing and factorization. We surveyed some existing knowledge tracing algorithms and applications, and some applications of factorization machine.

1.2.1 Property of high school Math

Disciplines and knowledge are closely related to each other, so that disciplinary knowledge denotes the specific knowledge contained in a particular field of study. Disciplines are referred to in this study only for specific subjects in the field of education, such as mathematics, language, chemistry and so on. The first step is to learn how to make the best use of the knowledge that is available. The knowledge is obtained from practice, so after learning it, it can also be applied to social practice. Scientific knowledge is declarative because it can be expressed in a series of symbols, words and diagrams; it is also procedural because it can be arranged and learned according to a specific logical order in the process of concrete learning.

Mathematics is a science specializing in the study of the relationship between quantities and spatial forms, its symbolic system is more complete, the formula structure is clear and unique, text and images and other expressions of language is also more vivid and intuitive.

The knowledge that learners need to learn mostly comes from the summaries of the experiences of their predecessors in practical activities. The learning process is a process of cognitive learning of the summarized knowledge and continuous digestion, adjustment and reorganization of the knowledge structure, so as to build a more perfect and suitable knowledge structure, as well as a process of integration with innovative thinking. Thus a good cognitive structure can promote the formation of knowledge structure, and a good knowledge structure can enrich the organization form of cognitive structure. Since the disciplinary knowledge structure consists of two parts: knowledge composition and knowledge dependency, we will analyze the disciplinary knowledge structure from these two aspects, knowledge structure and composition.

The knowledge that learners need to learn mostly comes from the summaries of the experiences of their predecessors in practical activities. The learning process is a process of cognitive learning of the summarized knowledge and continuous digestion, adjustment and reorganization of the knowledge structure, so as to build a more perfect and suitable knowledge structure, as well as a process of integration with innovative thinking. So a good cognitive structure can promote the formation of knowledge structure, and a good knowledge structure can enrich the organization form of cognitive structure. Since the disciplinary knowledge structure consists of two parts: knowledge composition and knowledge dependency, we will analyze the disciplinary knowledge structure from these two aspects.

Knowledge composition refers to the organization of knowledge within a subject area, which mainly includes knowledge points, knowledge blocks and knowledge systems.

- knowledge point: A point of knowledge is the smallest constituent unit of the knowledge structure of a discipline and is used to represent specific concepts.
- knowledge block: A knowledge block is a collection of one or more sets of knowledge points, also known as knowledge modules, in which knowledge blocks and knowledge blocks can be combined to form new knowledge modules, and a subset of knowledge blocks is called a knowledge sub-module.
- knowledge body: a body of knowledge is a structured system that is a combination of all the pieces of knowledge in a particular subject area.

Mathematics is a science that specializes in the relationship between quantity and spatial form. Its symbol system is more complete, the formula structure is clear and unique, and the language of expression such as words and images is more vivid and intuitive. The knowledge structure of senior secondary mathematics is a more logical and systematic knowledge system organized on the basis of the knowledge structure of junior secondary mathematics. This is because learning for any discipline needs to be based on the existing cognitive structure in order to progressively effective learning and skills training, so in the process of learning high school mathematics, you need to have a solid foundation of junior high school mathematics discipline knowledge. In the past few years, there have been a number of cases in which the students have been able to learn from each other.

- Highly abstract: Mathematics has a high degree of abstraction, because the discipline's knowledge system is built using many abstract knowledge concepts, and with the help of these concepts and knowledge to learn and expand thinking, forming new abstract conceptual knowledge. The abstraction of mathematics is reflected in the object is not concerned with the introduction of specific content, only the number of relationships between the spatial form. Therefore, abstraction in mathematics is different from abstraction in other disciplines in terms of both object and degree. There are also some differences between mathematics and the natural sciences, because in mathematics the accuracy of calculations, proofs, and inferences can only be verified using rigorous logical methods and cannot be tested by repetitive experiments, whereas in the natural sciences the verification is the opposite.

- Strict logic: The discipline of mathematics is very logical because any conclusion reached in mathematics requires rigorous logical reasoning and rigorous proof in order to be considered reasonable. However, mathematics is not the only discipline that possesses rigorous logic; other natural science studies of reasoning and proof must also possess a certain degree of logic. In mathematics, not all conclusions reached after reasoning and proof can be applied in practice, because many mathematical models are developed and mathematical conclusions drawn under ideal circumstances.

- Broad applicability: Mathematics is an important means and tool for us to participate in practical social activities or scientific research, and the study of mathematics is indispensable in all walks of life and in all areas of society. Therefore, mathematics has a wide range of applications and has become an important basis for the development of modern science.

1.2.2 Knowledge relation

Knowledge relations represent the connections between knowledge points (or between knowledge blocks and knowledge chunks) in the discipline knowledge structure. It is through these connections that different knowledge points can be formed into knowledge blocks, and different knowledge blocks can be combined to form the whole disciplinary network knowledge structure system. There are many different kinds of knowledge relationships, so that different definitions of knowledge relationships lead to different knowledge structures. Therefore, in order to unify the definition of knowledge relations, we divided them into general relations and special relations based on the general and special characteristics of discipline knowledge structure. The special relationships represent the unique knowledge relationships of a particular discipline, while the universal relationships represent the general relationships of any discipline. Secondly, according to the demands of knowledge graphping research, we divide universal relations into six kinds of knowledge relations: synonymous, fraternal, antecedent, consequent, inclusive and antagonistic; and special relations into four kinds of knowledge relations: detailed, transformative, causal and correlative.

- tautology: Expresses the relationship between two points of knowledge that have the same meaning as what is being described, e.g. regular and equilateral triangles.

- fraternity: Expresses the relationship between two knowledge points that have the same parent class.

- predecessor: It means that you need to finish learning knowledge point A before learning knowledge point B, that is, $A \rightarrow B$ is a precursor relationship.

- successor: denotes the inverse of the antecedent relationship, i.e., $B\,toA$ is the successor relationship.

- containment: Indicates that knowledge point B is included in the definition of knowledge point A, i.e., $A \rightarrow B$ is an inclusion relationship.

- antagonism: From a certain point of view, knowledge point A is incompatible with knowledge point B, i.e. $A \leftarrow B$ is an antagonistic relationship.

- refinement: A grammatical analysis of the definition of knowledge point A leads to knowledge point B, where $A \leftrightarrow B$ is a detailed relationship

- transformation: denotes that knowledge point A and knowledge point B can be transformed to each other under certain conditions, i.e., $A \leftrightarrow B$ is a transformation relationship.

- causation: denotes that knowledge point A can be deduced from knowledge point B as a known condition, i.e., $A \leftrightarrow B$ is a causal relationship.

- relation: Indicates that there is a relationship between the definitions of Knowledge Point A and Knowledge Point B, but the relationship is not explicitly specified, i.e., $A \leftrightarrow B$ are correlated.

In the process of constructing the discipline knowledge structure, firstly, we need to analyze the current discipline knowledge content, teaching objectives, teaching objects, teaching strategies and discipline characteristics in detail; secondly, we divide the whole discipline knowledge system into several knowledge modules, and then we divide each knowledge module into several knowledge points; finally, with reference to the above ten kinds of knowledge relationships and the knowledge relationships extracted from data sources, we can determine and establish the relationships between knowledge modules and knowledge modules, between knowledge modules and knowledge points, and between knowledge points and knowledge points, so as to form a complete discipline knowledge system structure.

1.2.3 Knowledge tracing algorithms

Knowledge Tracing is a technique that models students' knowledge acquisition based on their past answers to obtain a representation of their current knowledge state. The task is to automatically track the change of students' knowledge level over time based on their historical learning trajectory, in order to be able to accurately predict the students' performance in future learning and to provide appropriate learning tutoring. In this process, the knowledge space is used to describe the level of student knowledge acquisition. A knowledge space is a collection of concepts, and a student's mastery of a part of a collection of concepts constitutes the student's mastery of knowledge. Some educational researchers argue that students' mastery of a particular set of related knowledge points will affect their performance on the exercise, i.e., the set of knowledge that students have mastered is closely related to their external performance on the exercise.

There are several kinds of knowledge tracing algorithms:

- Bayesian knowledge tracing (BKT): Bayesian knowledge tracing is an early and commonly used knowledge tracing model, BKT uses user interaction modeling with real-time feedback to model a learner's potential knowledge state as a set of binary variables, each representing whether or not a knowledge point is understood, and there are dynamic changes in mastery of knowledge points as students continue to practice, BKT maintains binary variables of knowledge point proficiency by using Hidden Markov Models (HMM), the original BKT model does not take into account students' knowledge forgetting, and related studies address students' guesses, personal vivid knowledge mastery and problem difficulty factors on BKT[26].

- Deep Knowledge Tracing (DKT): The DKT model applies neural networks to the knowledge tracing task for the first time[20], using an LSTM model to track the dynamics of student knowledge proficiency over time, and to learn the potential vector representation of student knowledge proficiency directly from the data. The advantage of DKT is that it can record knowledge over a longer period of time based on students' recent answers. In addition, it can update the knowledge state based on each answer, only the last implicit state needs to be saved, no double counting is required, and it is suitable for online deployment. It does not require domain knowledge, works with any user answer dataset and automatically captures associations between similar questions. The disadvantage of DKT is that the model output fluctuates greatly when

the answer sequence is disrupted, i.e., the same questions and the same responses yield different knowledge states when the answer sequence is inconsistent. Due to the above-mentioned problems and the fact that students do not necessarily have continuous consistency in their knowledge during the answer process, it leads to bias in the prediction of students' knowledge states influenced by the sequence. There is also the black box problem, which sometimes leads to the strange situation that the first correct answer leads to a high prediction probability for all subsequent ones, while the first wrong answer leads to a low prediction probability for all subsequent ones.

- Dynamic Key-Value Memory Networks for Knowledge Tracing(DKVMN): Dynamic Key-Value Memory Networks for Knowledge Tracing (DKVMN) was proposed in 2017 by Jian[27]. Based on the strengths and weaknesses of BKT and DKT and using the memory augmentation neural network approach, the Dynamic Key-Value Memory Networks (DKVMN) is proposed. It borrows ideas from memory-enhanced neural networks and combines the advantages of BKT and DKT. DKVMN stores all knowledge points with a static matrix key and a dynamic matrix value to store and update the student's knowledge state. In the DKVMN paper, they compare DKVMN with DKT and a sophisticated version of BKT, BKT+. They found that DKVMN achieves excellent performance and is the most advanced model in the KT domain. In addition to improved performance, it has several other advantages over LSTM, including prevention of overfitting, a smaller number of parameters, and automatic discovery of similar practice questions by underlying concepts. In addition, Chaudhry[4] improves the performance of DKVMN by jointly training request cue prediction with knowledge tracing through multi-task learning.

1.3 Research Objectives and Content

The purpose of this study is to build a high school mathematics learning resource recommendation system based on knowledge tracing and factorization machine algorithm. We use knowledge tracing to model students' knowledge states, which outputs a graphical knowledge state vector, which we use as the next-level input, considering students' individualized differences and knowledge forgetting process, and apply the factorization machine algorithm to the resource recommendation system. For knowledge tracing, we build a graph neural network-based knowledge tracing model, which can well characterize

the intrinsic connections of knowledge points in mathematics subjects considering that the knowledge points are a graph-like structure, and output a graph knowledge vector matrix, which can also effectively characterize the connections between problems and knowledge points. The output of the knowledge tracing model is then passed through a factorization machine algorithm to obtain the recommendation degree of the learning resources and output a vector of recommendation weights for different learning resources.

1.4 Thesis Organization and Structure

Chapter 1 of this paper is an introduction. It introduces the research background of the study, current industry-related research progress and the focus of the study. Then it leads to the three core points of this paper: learning resource representation, knowledge tracing and resource recommendation.

Chapter 2 of this paper concentrates on learning resource representation, which addresses storing learning resources through knowledge graphs. This paper explores some concepts of knowledge graphs, related studies, and then gives the process of knowledge graph building. It is also demonstrated that knowledge graph building can effectively characterize the a priori intrinsic features of subject knowledge.

Chapter 3 of this paper gives a knowledge tracing model of pre-trained graph neural network, which better characterizes the graph-like properties of knowledge. It is able to transform the knowledge tracing task into a time-series node-level classification problem in GNNs. Since the knowledge graph structure is not explicitly provided in most cases, we present various implementations of the graph structure. Empirical tests on two open datasets show that the method improves the prediction of student performance without any additional information and shows more interpretable predictions. The inclusion of pre-training is also attempted during the experiments, which can greatly improve the training efficiency and performance.

Chapter 4 of this paper proposes the application of a deep factorization machine algorithm with knowledge tracing model data as input to build a learning resource recommendation system. The output is a weight vector of top n, characterizing the recommended resources of top n.

Chapter 5 of this paper presents the conclusion.

Chapter 2

Exercise Knowledge Point Mining Based on Graph Neural Network

2.1 Research Motivation

In all aspects of the whole recommendation system, including data collection, data mining and data recommendation, the construction of the recommendation data source is the most important cornerstone. Establishing an accurate and complete question bank is the first and important step to build a test recommendation system. The reasonableness of the overall design of the question bank, as well as the quality, is the key to the success or failure of the question bank recommendation system. Question bank construction is also a very complex and difficult work, both hard work and the understanding of education, technical ability to grasp a very high demand of work. In the process of building the question bank is not simply the accumulation of quantity, if not take into account regional differences, different test-taking requirements and effectiveness and other factors. High school mathematics subject has thousands of scale of knowledge points, on average, each knowledge point has dozens of exercises from easy to difficult of different difficulty, a quality mathematics subject question bank of 100,000 scale. In addition to the quantitative requirements, there are two fundamental issues for a quality question bank besides the quality of questions and the matching of educational contents (textbooks): one is the construction of knowledge point relationship network, and the other is the construction of knowledge point-exercise relationship. It is the optimization of these factors that leads to the construction threshold of a quality question bank, and the extremely high cost. The question databases on the market

often lack the attention to these factors, resulting in many poor-quality exercise databases. Therefore, the labeling of knowledge points of the topics and the construction of knowledge system is one of the most core issues in the process of building a high-quality question database.

One of the key problems in building a test recommendation system is how to recommend topics in conjunction with knowledge points. When considering a solution to this problem, the first thing we need to consider is what kind of knowledge system and knowledge point labels we need. This is not a very easy question to answer. First of all, there are various descriptions of "knowledge points" in textbooks and syllabi, as well as in various teaching aids. In high school mathematics, knowledge points, such as functions, definition domains, value domains, or analytic equations, have direct concepts, applications of methods, abstractions of topics, clever summaries of similar solutions, etc.-in short, there is no standard system of names, connotations, granularity, hierarchy, or even so-called relationships or connections between knowledge points. The names, connotations, granularity, hierarchy, and even the so-called relationships or connections between knowledge points can vary widely from one source to another. Both the high level of educational experts and the flexible experience of frontline teachers are needed here. Second, as a data mining task, we are concerned with - in terms of the set of tags - whether it is a closed set or an open set, and we are concerned with whether a "knowledge point" tag is a categorical concept or a keyword (or a combination of keywords, phrases). combination of keywords, phrase pairing. The clarity of this definition is crucial to the problem definition of data mining and basically determines the choice of the solution that follows. However, this clarity should not only consider the pedagogical significance, but also the way the product uses the knowledge point label, especially the latter, which may be a more important factor.

As a system for students to push questions for self-study and visualize the analysis report, then the tags of knowledge points 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, so as to build a more powerful user model and recommendation engine, while the report is also convenient for targeted analysis and understanding, this time the classification system and tag collection need to be taken, the granularity of its knowledge points should be fine enough, and the level of the knowledge system will be much larger, at the same time, "knowledge points" even if it is not a standard system, it needs to be recognized by most teachers and students or a certain degree of use.

In this chapter, the knowledge point tags are mined for topic recommendation and analysis reports. He requires to improve the knowledge point association of the topic, i.e., to label the topic with a number of corresponding knowledge points, among which there will also be dependencies between the knowledge points, in addition to establishing a complete knowledge map of knowledge points for generating student learning reports. For the first problem, it is actually a knowledge point mining task using the topic text, which is also a classification task. To be more specific, it is a task of hierarchical classification based on the information in the short text of the topic. In this task, we need to use the most basic technologies of natural language processing and machine learning. That is, 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 natural language processing techniques and obtaining the classification model through machine learning - the system The system has the ability to 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 of the learning system is a set of training corpus, i.e., n questions that have been labeled and their corresponding knowledge point labels; based on this data experience, the learning system trains (or learns) a model, which can be a functional expression $y = f(x)$ or a probabilistic expression $p(y|x)$, and this is the output of the learning system. Based on the classification model of the learning system, the classification system can then make predictions, i.e., a new topic is entered and the classification system automatically identifies its most likely knowledge labels. The second problem is then to construct a knowledge map of high school mathematics knowledge points, which shows the structure and association of high school mathematics knowledge points. Knowledge resources and their carriers are described through visualization techniques to mine, analyze, construct, map, and display knowledge and its interconnections. Meanwhile knowledge graphs, as an important part of artificial intelligence, are now playing an increasingly important role in recommendation systems and question and answer systems. In addition combining the relationship of high school mathematics knowledge points and students' learning state model, it can also better realize the adaptive planning of learning paths.

2.2 Proposed Model

In this section, a graph neural network-based model is proposed for test question-knowledge point association and knowledge mapping of knowledge points. The model is roughly divided into two parts. The first part is a test-knowledge point association module implemented with a semi-supervised learning algorithm based on graph convolutional neural networks and text mining embedding learning for test knowledge point labeling and association. The second part considers the systematization of knowledge of high school mathematics and combines a priori domain knowledge with an improved R-GCN algorithm to construct a knowledge graph of high school mathematics knowledge points.

2.2.1 Model Overview

The task of this section is to construct a model for mining the exercise-knowledge point relationship and a model for mining the association between knowledge points. Establishing an exercise-knowledge point relationship means labeling the knowledge points of an exercise, which is a collection of knowledge concepts used to understand and solve the exercise. Therefore, accurately describing the knowledge points of a test question is important 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 the knowledge points of test questions, but when the number of questions or the complexity of the questions is high, manual labeling has problems such as high workload, high subjectivity and imperfect labeling. In addition, considering the association between knowledge points, manual annotation also suffers from the inability to take into account the knowledge interconnection relationship. Another way is to use machine learning or deep learning to automatically annotate the knowledge points of the exercises. Because only a small number of exercises in the exercise set are labeled with knowledge points, while a large number of other exercises are not yet labeled, these knowledge point labeling is often a semi-supervised or unsupervised learning mode. An exercise often has multiple knowledge points, so in effect knowledge point mining is a multi-label classification problem [23, 29, 15]. This chapter also discusses “how to effectively model the relationship between knowledge points” and proposes a multi-label classification model based on graph neural networks. In the knowledge extraction part of the exercises, the textual information of the exercises is often mined to automatically label

the knowledge points. In order to improve the feature extraction performance, Multimodal Representation [11] can also be considered to extract the illustrations of the exercises, mathematical formulas, etc. as additional features.

In 2019, a multi-label image classification model was proposed[5], and inspired by this model, this chapter proposes a multi-knowledge point labeling model based on attention mechanism exercise text feature extraction and graph neural network-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 as follows Fig.2.1.

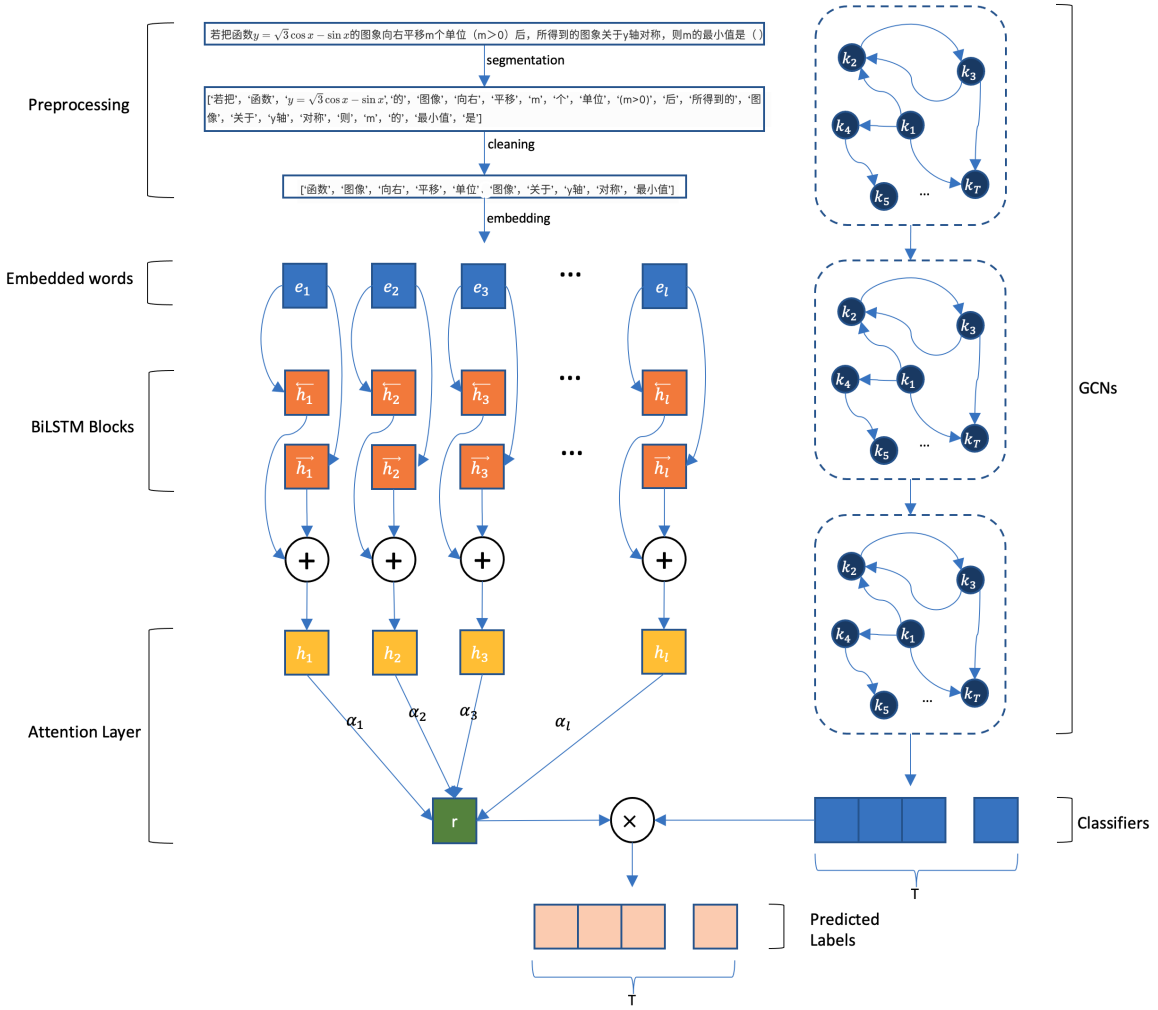


Fig. 2.1 Structure of the Knowledge Tagging Model

From the structure we can see that there are generally two parts in the model:

1. The first part is the exercise description text mining module, which uses natural language processing techniques to text mine the questions (including question descriptions and answers), and it contains hidden knowledge information. In this paper, an end-to-end network training approach is designed to achieve the overall iterative learning of the model. Specifically, it includes a text preprocessing part that performs the text segmentation, filtering and de-duplication parts, an embedding layer that performs the word vector embedding calculation and an Attention-based bidirectional LSTM network that performs text information mining, which was proposed by Peng et al. in 2016[30], outputs a vector representation for the text of the exercise.

2. The second part is a graph convolutional neural network based knowledge point association multi-label classifier, which maps knowledge points to a set of interdependent target classifiers. These classifiers are computed with the vector of exercises outputted in the first part to achieve end-to-end training of the whole network.

2.2.2 The Exercise Description Text Mining

This section is based on the Attention based Bi-LSTM model proposed by Peng et al. The structure of this model is shown in the Fig.2.2.

The model include 4 layers: Pre-process Layer, Embedding Layer, Bidirectional LSTM layer, Attention Layer and Output Layer:

Pre-process Layer

In the preprocessing stage, it mainly includes word segmentation, cleaning, and other optional steps. Considering that our research object is a Chinese high school math test, compared with English, there are no intermediate spaces in Chinese sentences, so a word segmentation algorithm must be used to break the sentence into word segmentation. There are many texts in the content that are irrelevant to the meaning of the sentence. If you directly calculate it, it will cause a lot of interference and redundant information, so additional text cleaning is also a necessary step. The Fig.2.3 shows an example of preprocessing of an exercise text.

- Segmentation: Chinese word segmentation is a basic step of Chinese natural language processing. In Chinese, there is no natural separation between words in a sentence, so the sentence must be segmented first to break the sentence into words. The effect

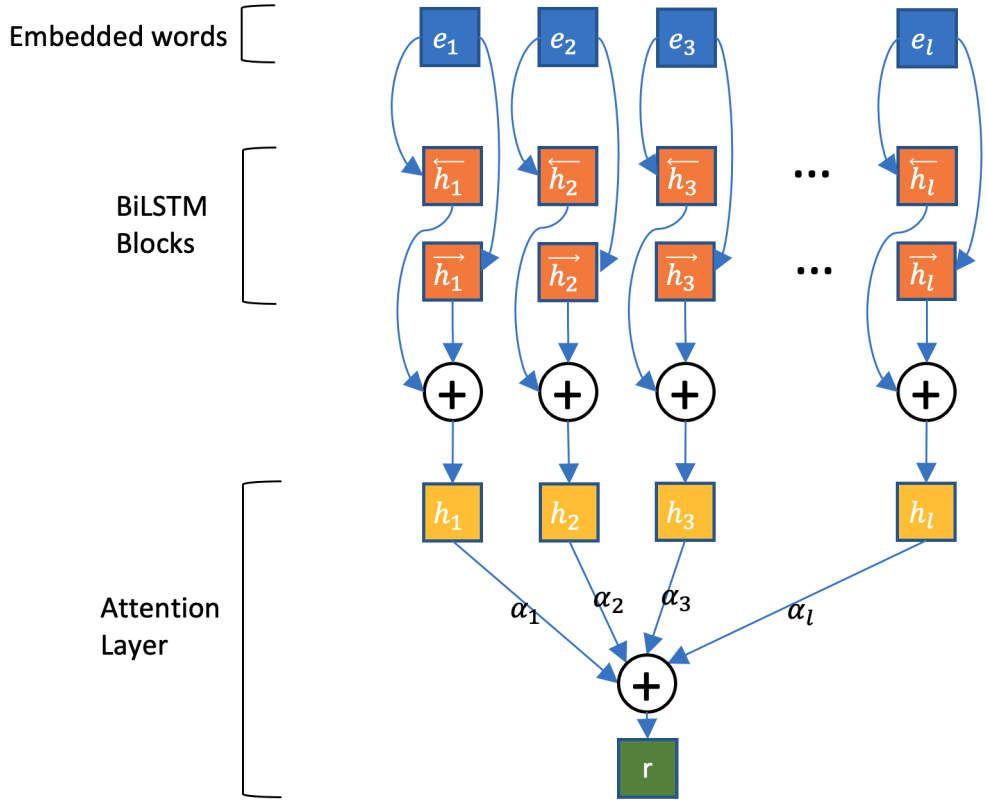


Fig. 2.2 Structure of Attention Based Bi-LSTM

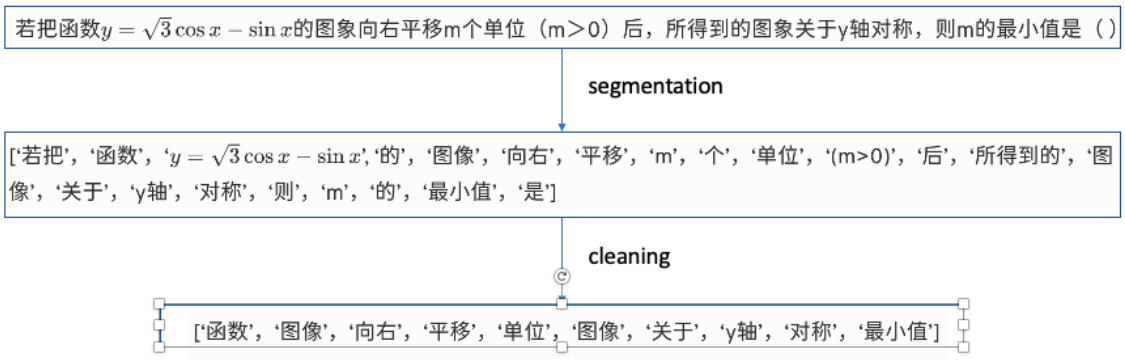


Fig. 2.3 Example of Preprocessing

of word segmentation will directly affect the effect of parts of speech, syntax tree and other modules. Choosing an appropriate Chinese word segmentation algorithm can achieve better natural language processing effects and help computers understand complex Chinese language. At present, Chinese word segmentation is mainly divided into two algorithms based on dictionary rule matching and based on statistical model. Compared with the former, the latter has better generalization and learning ability. It is used for word segmentation outside the prior rules such as ambiguous words and unregistered Words perform better. In this model, the current popular jieba word segmentation is used, which realizes word map scanning based on the Trie book structure, and uses algorithms such as stage planning and HMM model to find the largest word frequency-based segmentation grouping and merge to realize the future Recognition of login words. Users can also customize stop words and user dictionaries to realize 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 in a sentence, that is, stop words, their existence does not affect the meaning of the sentence. In the text, there will be a large number of function words, pronouns, or verbs and nouns with no specific meaning. These words are not helpful to the text analysis, so these stop words can be removed. For the exercise text, there are many mathematical expressions, symbols, etc., considering that these expressions in many exercises are not in text format, and OCR technology must be used to preprocess mathematical expressions from pictures to text. Therefore, when the Chinese text is sufficient, you can consider filtering out mathematical expressions to reduce the calculation load.

After the steps of data processing, a clean text token sequence is obtained, and then the word2vec technology can be used to perform text embedding operations on the Embedding layer.

Embedding Layer

In the application of deep learning, Embedding, a way of transforming discrete variables into continuous vectors, has greatly expanded the application of neural networks in various aspects. Embedding is a way to convert discrete variables into continuous vector

representation. In neural networks, embedding is very useful, because it can not only reduce the spatial dimension of discrete variables, but also represent the variables meaningfully. For example, in natural language processing, if simple one-hot encoding is used, it will often cause too many dimensions and sparse vectors, and it is impossible to learn the dependencies between vectors. In the process of embedding training, the embedded vector will be updated, which can clearly show the practice between vectors. In 2013, Mikolov et al. proposed an improved CBOW (Continuous Bag-of-Words) and Skip-gram model based on Neural Network Language Model (NNLM)[3], Recurrent Neural Network Language Model (RNNLM) [17] and C&W models[16], and in the next few years it became a classic solution for word embedding, the model is like Fig.2.4. In another paper by Mikolov in the same year, two strategies for training the Skip-gram model: Hierarchical Softmax and Negative Sampling were proposed[18]. The principle of CBOW is to predict the word itself based on several words around the input (that is, predict the word through context), while the Skip-gram model predicts several surrounding words based on the input word (that is, the input word predicts the context), like following formula 2.1–2.2.

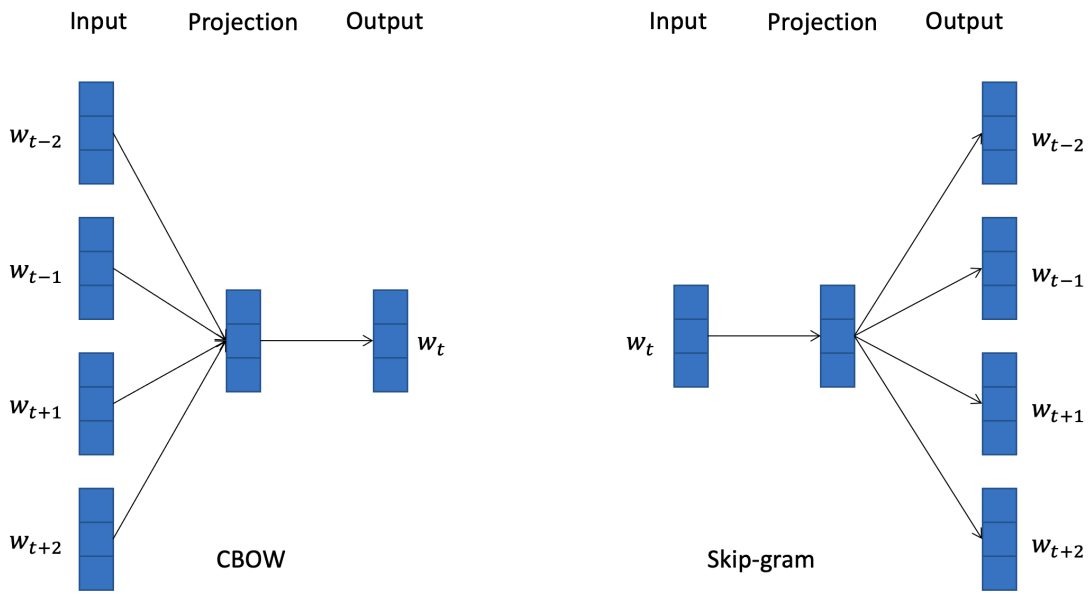


Fig. 2.4 Method of CBOW and Skip-gram

$$\text{CBOW: context}\ (w_t) = (w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}) \overset{\text{predict}}{\longrightarrow} w_t \tag{2.1}$$

$$\text{skip-gram: } w_t \overset{\text{predict}}{\longrightarrow} \text{context}\ (w_t) = (w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}) \tag{2.2}$$

Where c is the range of predicting words, w_i represents the ith word and $context(w_i)$ represents the context of ith word.

In this layer, we obtain a word token sequence extracted from the original exercise text. Set the sequence as $S = \{w_1, w_2, \ldots, w_l\}$, each w_i represents a word, and the sequence length is l. Set S as the size of non-embedded one-hot vector v_i representing w_i, i.e. $v_i \in \mathbb{R}^S$. After embedding operation, each token is transformed into an embedded vector e_i, that is, $e_i = \mathscr{E}(w_i)$, where $\mathscr{E}(\cdot)$ is the embedding function, $e_i \in \mathbb{R}^{d^e}$ and d^e is the dimension of the embedded vector.In fact, the embedding function can be modeled by a matrix multiplication: $\mathscr{E}(\cdot) = W^w \times \cdot$, where the learnable parameter $W^w \in \mathbb{R}^{d^e \times S}$ is the embedding transition matrix. So

$$e_i = W^w v_i \tag{2.3}$$

Bidirectional LSTM Layer

In order to solve the problem of text sequence learning, recurrent neural networks are generally used to process sequence data. Compared with the general neural network, it can process the data of the sequence change. For example, the meaning of a word will have different meanings because of the different content mentioned above, and RNN can solve this kind of problem well. In this section, the core of the task is a sequence-to-sequence (Seq2Seq) task. Input a title description embedding vector, and output a sequence containing the current training information. The simplest idea is to use the original RNN, whose structure is shown in the Fig.2.5. x_t is the input of data in the state t, and h represents the input of the previous node received. y_t is the output under the current node state t, and h^t is the output passed to the next node. So

$$h^{t+1} = f(W_t^h h^t + W^i x^t) \tag{2.4}$$

$$y^{t+1} = f(W^o h^{t+1}) \tag{2.5}$$

where σ is linear activation function like sigmoid, ReLU, etc.

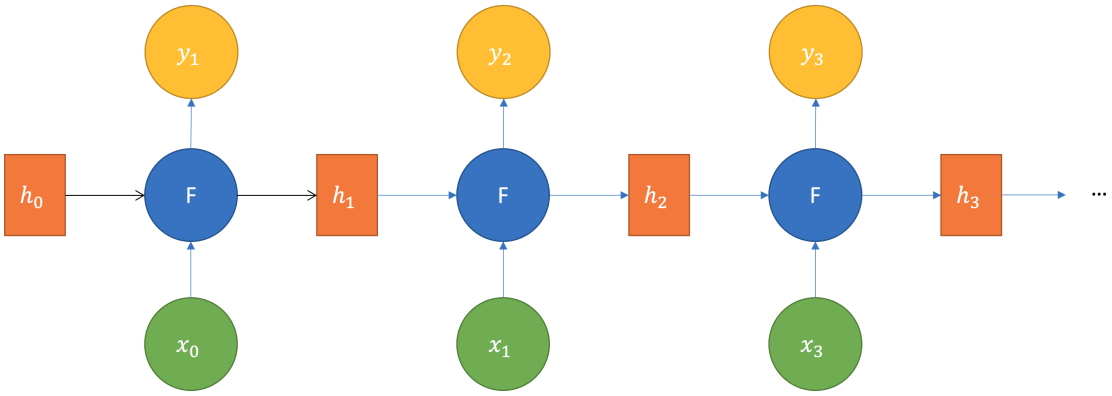


Fig. 2.5 Naive RNN Unit

One of the key points of RNNs is that they can be used to connect previous information to the current task. But when the sequence is longer, the problem of long dependency occurs. For example, for a sequence, the current state depends on a state far away from the current state, and the RNNs' ability to learn this state degrades greatly as the interval increases.

Long short-term memory (Long short-term memory, LSTM) is a special RNN, mainly to solve the problem of gradient disappearance and gradient explosion in the training process of long sequences. It was proposed by Hochreiter and Schmidhuber in 1997[12], and was improved and promoted by many following work. LSTM has performed very well on a variety of problems and is now widely used. The general model of LSTM is like Fig.2.6.

Compared with RNN which has only one transmission state h^t, LSTM has two transmission states, one cell state C_t, and one hidden state h_t. The cell state in LSTM corresponds to the role of the hidden state in RNN. Among them, the cell state C_t passed down changes very slowly while the hidden state h_t often has a big difference under different nodes. LSTM has the ability to remove or add information to the cell state through a well-designed structure called a "gate". A door is a way of letting information through selectively. They include a sigmoid neural network layer and a point-wise multiplication operation. The Sigmoid layer outputs a value between 0 and 1, describing how much volume can pass through each part. 0 means "no amount is allowed to pass", 1 means "allow any amount to pass". LSTM has three gates to protect and control the cell state.

The core part of LSTM is the connection between cell states, which is generally called the cell state. The reason why LSTM can solve the long-term dependence of RNN is because

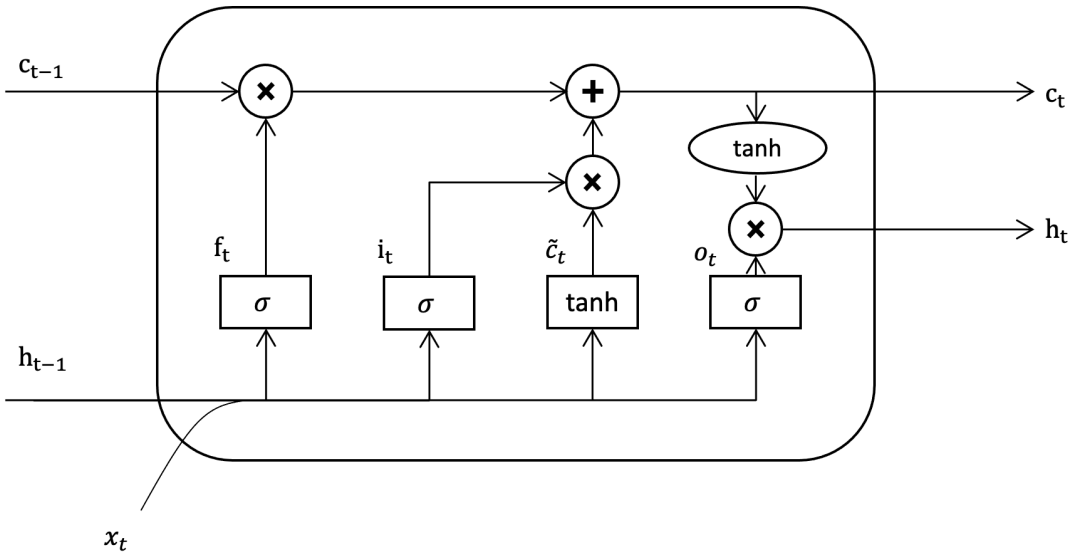


Fig. 2.6 Structure of LSTM Unit

LSTM introduces a gate mechanism to control the circulation and loss of features. The three kind of gate in LSTM is forget gate, input gate and output gate:

- **Forget Gate**: The first step in the LSTM is to decide what information to discard from the cell state. This decision is done through a gate called the forgetting gate. The gate reads h_{t-1} and x_t and outputs a value between 0 and 1 to each number in the cell state C_{t-1}. 1 means "keep completely" and 0 means "discard completely". The calculation formula is 2.6:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.6}$$

  The LSTM network determines what percentage of the previous content the network will remember through learning. The W_f and b_f is learnable parameter of forget gate.

- **Input Gate**: The next step is to determine what new information is stored in the cell state. There are two parts here. First, the sigmoid layer called the "input gate layer" determines what values will be updated. Then, a tanh layer creates a new candidate

value vector $\widetilde{C}_t$, which will be added to the state. The calculation formula is 2.7–2.8:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.7}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.8}$$

$\tilde{C}_t$ represents the update value of the unit state, which is obtained from the input data x_t and hidden nodes h_{t-1} through a neural network layer. The activation function of the update value of the unit state usually uses tanh. i_t is called an input gate, which is also a vector with elements in the interval of $[0,1]$ just like f_t, which is also calculated by x_t and h_{t-1} through *sigmoid*activation function. W_i, W_C, b_i, b_C are the learnable parameters of the input gate. i_t is used to control which features of $\tilde{C}_t$ are used to update C_t, following the same method as f_t, like formula 2.9:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{2.9}$$

- **Output gate**: The final step is to determine the output value. This output will be based on the cell state of the network. First, we run a sigmoid layer to determine which part of the cell state will be output. Next, we process the cell state through tanh (get a value between -1 and 1) and multiply it with the output of the sigmoid gate, and finally output the part that determines the output. The calculation is like formula 2.10–2.11:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.10}$$

$$h_t = o_t * \tanh(C_t) \tag{2.11}$$

The W_o and b_o are learnable parameters of output gate.

However, there is still a problem with using one-way LSTM to model sentences: it is impossible to encode information from back to front. In more fine-grained classification, such as the five classification tasks for strong commendation, weak commendation, neutral, weak derogation, and strong derogation, we need to pay attention to the interaction between emotion words, degree words, and negative words. Two-way LSTM can better capture the two-way semantic dependence. Bidirectional LSTM (Bi-LSTM) can better capture the two-way semantic dependence. 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 element-wise addition. The output of bidirectional LSTM is:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(e_t) \tag{2.12}$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(e_t) \tag{2.13}$$

$$h_t = \overrightarrow{h_t} \bigoplus \overleftarrow{h_t} \tag{2.14}$$

The Bi-LSTM Structure is like Fig.2.7.

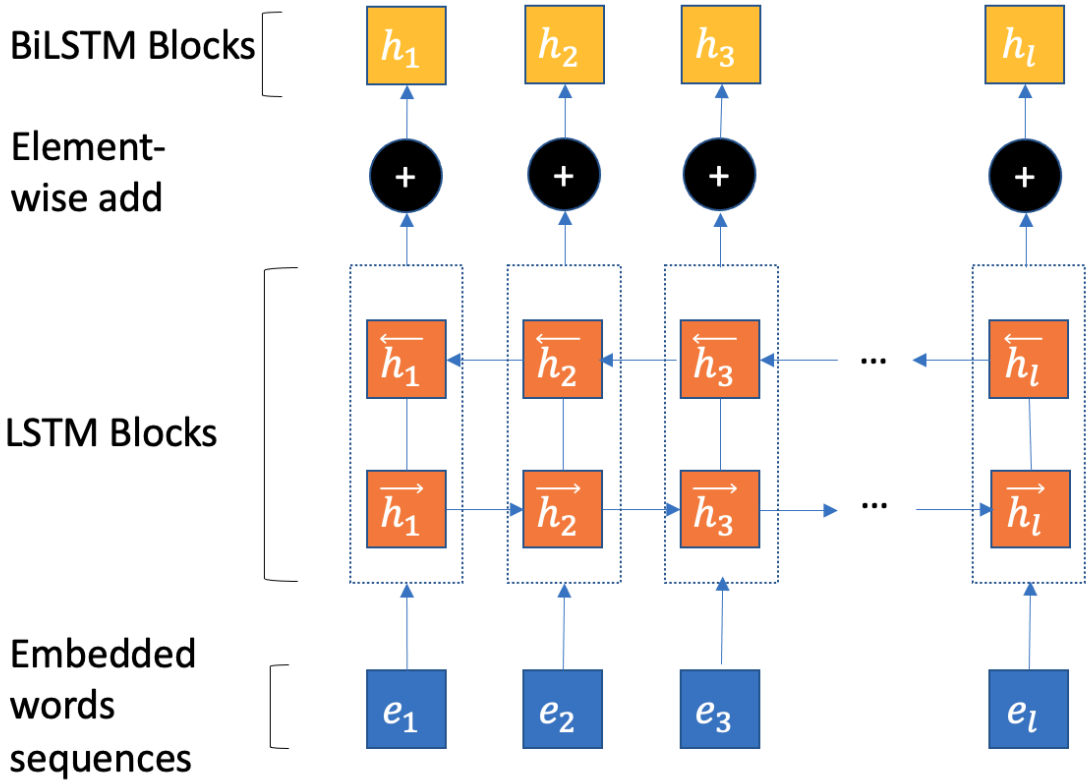


Fig. 2.7 Structure of Bi-LSTM

Here, the Bi-LSTM output a bidirectional sequence as the sum of element-wise addition between positive-order and reverse-order LSTM output sequence.

Attention Layer

In this model, the entire text mining module is actually an encoder-decoder framework. The encoding stage encodes the embedding representation of the word vector and encodes it

into a hidden semantic vector, and then the decoder decodes the semantic vector and outputs an output sequence. The limitation of this framework is that for a fixed-length semantic vector, the information of the entire sequence cannot be represented, and the information first input to the network will be covered by subsequent information, and as the sequence increases, the coverage and loss of information will get worse.

In order to solve the drawbacks in the encoder-decoder framework, the Attention mechanism was proposed by Bahdanau et al.[2]. Human attention is a mechanism for quickly screening high-value information from a large amount of information. The deep learning attention mechanism is inspired by the human attention mechanism. This method is widely used in various types of deep learning tasks such as natural language processing[13], image classification[10, 22], and speech recognition[6], and has achieved remarkable results.

Actually, The essence of the attention mechanism is a series of key-value pairs <K,V> contained in Source S, given an element of Query Q, calculate the similarity between Q and each K, and remember the weight coefficient of the corresponding value V. It is showed in Fig.2.8. Then a weighted sum is performed to obtain the final Attention value. It can be expressed as:

$$Att(Q,S) = \sum_{i=1}^{|S|} Sim(Q,K_i) * V_i \tag{2.15}$$

Where Att and Sim represent attention and similarity.

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 principle is that not all words have the same effect on the meaning of sentences. Therefore, we introduce an attention mechanism to extract words that are important to the meaning of a sentence, and summarize the representations of those information words to form a sentence vector. Specifically:

$$u_t = \tanh(W_\omega h_t + b_\omega) \tag{2.16}$$

$$\alpha_t = Softmax(u_t^T u_\omega) = \frac{\exp(u_t^T u_\omega)}{\sum_t \exp(u_t^T u_\omega)} \tag{2.17}$$

$$r = \sum_t \alpha_t h_t \tag{2.18}$$

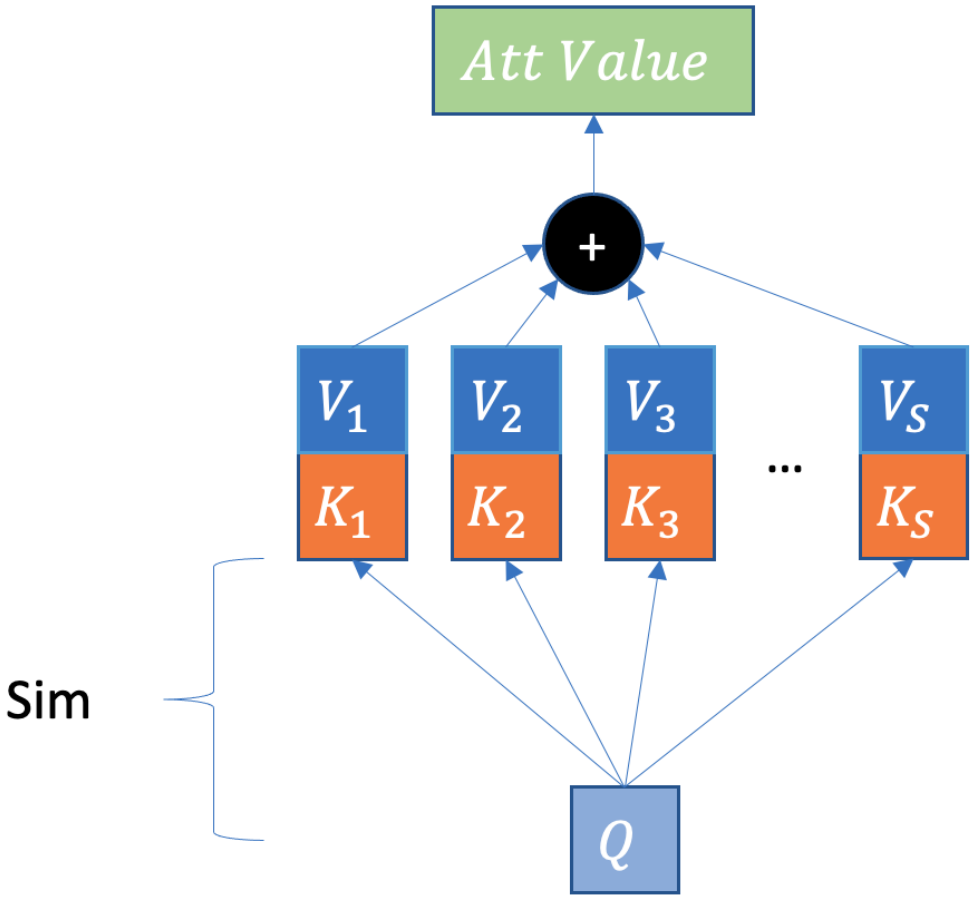


Fig. 2.8 The essential idea of Attention mechanism

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 word-level context vector. By measuring the similarity between u_t and u_ω as the importance of the word, and normalize it with the softmax function to obtain the importance weight α_t. After that, the entire text is represented as a weighted sum of word annotations based on weights. 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 the training.

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 and so on. 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 the task of this model, it gives descriptions and answers to exercises, some of which have already marked knowledge points, and use this information to mark knowledge points for exercises that are labeled knowledge points. Taking into account the dependence between knowledge points, some deep 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 graph convolutional neural network is used to model the knowledge point relationship graph, and each knowledge point corresponds to a node on the graph. After multiple graph convolution calculations, a series of classifiers are generated. These classifiers respectively act on the text vector generated by the text mining module. Each classifier outputs a value representing the probability that the knowledge point is associated with the exercise. Its overall structure is shown in the Fig.2.9.

In the proposed GCN-based model, knowledge point labels are represented by word embedding. Knowledge point set $K = \{k_1, k_2, \ldots, k_T\}$, where T is the total number of knowledge points. For each single knowledge point k_i, $k_i \in \mathbb{R}^{T \times d_{ko}}$, d_{ko} is the dimensionality of the embedding vector of knowledge point object. In this paper, a learnable stacked GCN network is used to transform these knowledge point objects one by one into an internally connected knowledge point object classifier $C = [c_1, c_2, \ldots, c_n]$, where $c_i \in \mathbb{R}^{T \times d^r}$, d^r is the dimensionality of the text representation vector r output by the text mining module. These

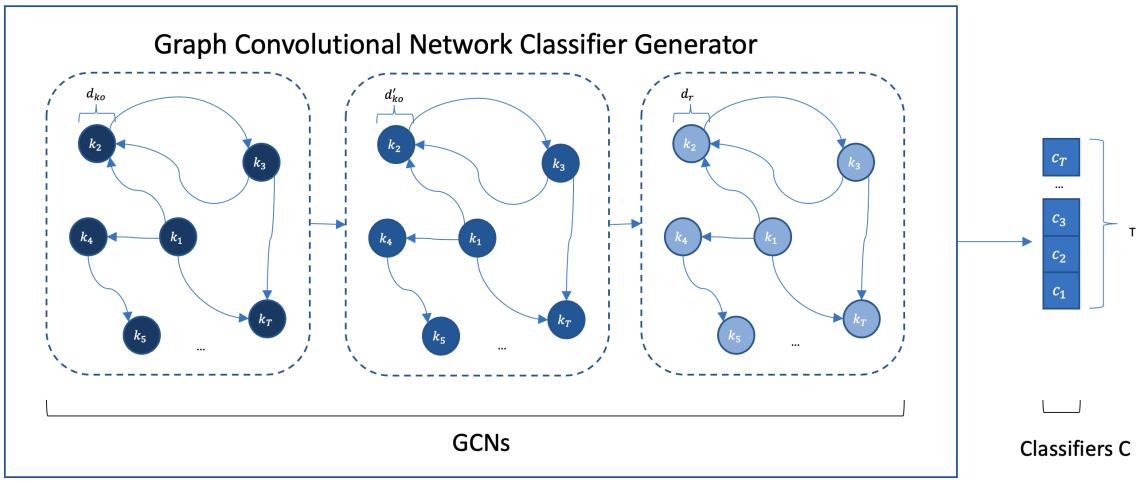


Fig. 2.9 The structure of GCN Classifier Generator

classifiers and r can be used to calculate the dot product of each label. The probability of the label.

The reason why this model adopts the GCN structure is that 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, and calculate the co-relation matrix by calculating the co-occurrence of knowledge points in the exercises.

Graph Convolutinal Network

In the real world, many important data sets exist in the form of graphs or networks, such as social networks, knowledge graphs, protein interaction networks, world trade networks, and so on. Some papers reviewed this problem and tried to generalize neural networks and apply them to arbitrary graph structure data[25, 8]. Currently, most graph neural network models have a common architecture. Known as graph convolutional neural networks (GCNs), these models are convolutional because the filter parameters can be shared at all positions or a local position in the graph. For these models, their goal is to learn a mapping of signals or features on the graph $G = (V, E)$. Their inputs include:

- The feature description x_i of each node i can be written as an $N \times D$ feature matrix (N represents the number of nodes, D represents the number of input features)

- Characteristic description of graph structure in matrix form, usually in the form of adjacency matrix, denoted as A

The model will produce a node-level output Z (an $N \times F$ feature matrix, where F represents the number of output features of each node). Graph-level output can be modeled by introducing some pooling operations.

Each neural network layer can be written as a nonlinear function:

$$H^{(l+1)} = f(H^{(l)}, A) \tag{2.19}$$

$$H^{(0)} = X \tag{2.20}$$

$$H^{(L)} = Z \tag{2.21}$$

L is the number of layers. The core problem is how the f is selected and parameterized. The f can be represented as 2.22 by applying the convolutional operation[14]:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2.22}$$

the input of the l-th layer network is $H^{(l)} \in \mathbb{R}^{N \times D}$ (the initial input is $H^{(0)}$), N is the number of nodes in the graph, and each node is represented by a D-dimensional feature vector. $\tilde{A} = A + I_N$ is the adjacency matrix with self-connection added, $\tilde{D}$ is the degree matrix, $\tilde{D_{ii}} = \sum_j \tilde{A}_{ij}$. $W^{(l)} \in \mathbb{R}^{D \times D}$ is the parameter to be trained. σ is the corresponding activation function, such as ReLU.

Multi-label Recognition

From the stacked GCN network, we can learn interdependent object classifiers $C = \{c_1, c_2, \ldots, c_T\}$, where T represents the number of categories. For the input layer, the input is $r \in \mathbb{R}^{T \times d_{ko}}$, where d_{ko} is the number of dimensions represented by the embedding of the knowledge point object. Each GCN layer 1 takes the node representation from the previous layer $H^{(l)}$ as input and outputs a new node representation $H^{(l+1)}$. The output layer of the last layer is C, where d_r represents the dimension of the title text. Finally, the label prediction vector $\hat{y}$ can be obtained by the dot product product of the learned classifier and the title text

representation r:

$$\hat{y} = Cr \tag{2.23}$$

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. Through end-to-end training, the loss function $\mathscr{L}$ can be written as:

$$\mathscr{L} = \sum_{i=1}^{T} y_i \log(\text{sigmoid}(\hat{y}_i)) + (1 - y_i) \log(1 - \text{sigmoid}(\hat{y}_i)) \tag{2.24}$$

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 key step in the GCN model. In this model, the commonly used data association rule mining algorithm Apriori algorithm is used to calculate the knowledge point association by counting the number of occurrences of the knowledge point set.

In this model, in fact, 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 L_i appearing, another label L_j appears, or the conditional probability of the data. Lift represents the probability that the label L_i is contained at the same time, and the ratio of the probability of occurrence of X population:

$$\text{Support}(L_i, L_j) = P(L_i, L_j) = \frac{\text{number}(L_i, L_j)}{\text{number}(\text{ All Samples })} \tag{2.25}$$

$$\text{Confidence}(L_i arrow L_j) = P(L_i \mid L_j) = P(L_i, L_j)/P(L_j) \tag{2.26}$$

$$\text{Lift}(L_i arrow L_j) = P(L_i \mid L_j)/P(L_i) = \text{Confidence}(L_i arrow L_j)/P(L_i) \tag{2.27}$$

Similar to calculating Support, the frequency matrix $E \in \mathbb{R}^{T \times T}$ of the sample knowledge point label pairs in the exercise training set can be calculated here, where T is the number of knowledge points. M_{ij} represents the number of times the knowledge point i and the

knowledge point j appear at the same time. Similarly, the knowledge point pair can be calculated by calculating Confidence to calculate the conditional probability matrix P, where $P_{ij} = P(L_i, L_j)/P(L_j)$, where $P_{ij} = P(L_i, L_j)/P(L_j)$ means the situation when the knowledge point j appears The conditional probability of the occurrence of the next knowledge point i.

Similar to calculating Support, the frequency matrix $E \in \mathbb{R}^{T \times T}$ of the sample knowledge point label pairs in the exercise training set can be calculated here, where T is the number of knowledge points. E_{ij} represents the number of times the knowledge point i and the knowledge point j appear at the same time. Similarly, knowledge point pairs can be calculated by calculating Confidence to calculate the conditional probability matrix P, where $P_{ij} = P(L_i, L_j)/P(L_j)$ means the situation when the knowledge point j appears The conditional probability of the occurrence of the next 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 filter the noise caused by the label pairs of rare knowledge points, a minimum knowledge confidence threshold can be set. When P_{ij} is greater than the given threshold τ, naming activating value, then A_{ij} is set to P_{ij}, Otherwise A_{ij} is set to 0. The formula is like 2.28.

$$A_{ij} = \begin{cases} 0, & \text{if } P_{ij} < \tau \\ P_{ij}, & \text{if } P_{ij} \geq \tau \end{cases} \tag{2.28}$$

2.3 Experiment

This chapter proposes a multi-label labeling model for exercise knowledge points. In 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 in this article comes from the labeled college entrance examination math test questions and simulation questions (including answer analysis) on the online website koolearn.com. 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.2.10.

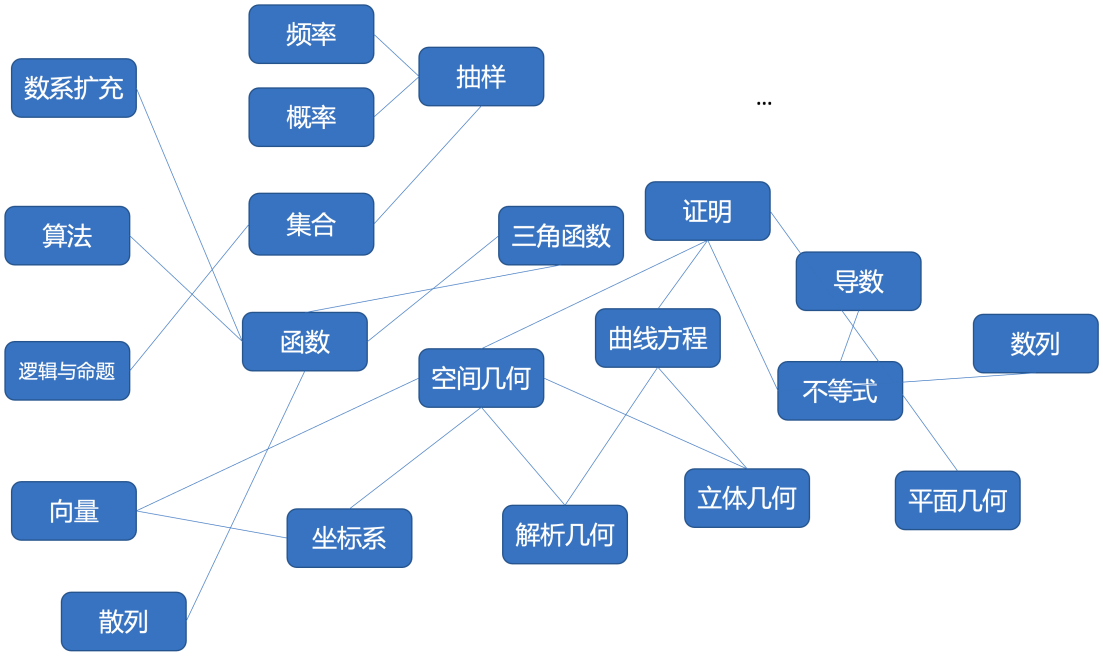


Fig. 2.10 Knowledge Points of dataset

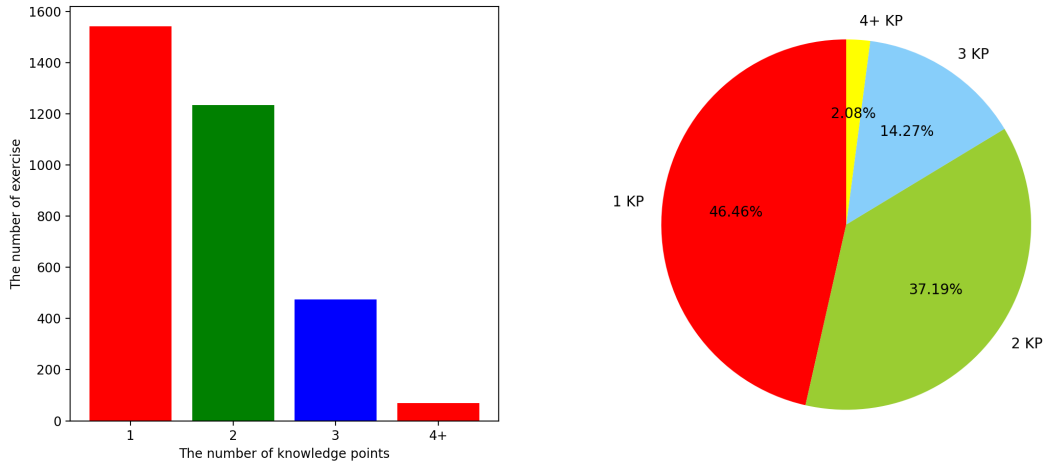
The distribution of the number of exercise knowledge points in this data set is shown in the Fig.2.11.

2.3.2 Baseline

At present, there are already some algorithm models for labeling text. In order to verify the effectiveness of the algorithm proposed in this paper, it will be compared with the following algorithm as a baseline indicator.

- 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.[28], it find the k samples closest to the sample through the KNN algorithm, count the number of each category



(a) Number of Exercises (b) Portion of Exercises

Fig. 2.11 Distribution of the number of knowledge points of exercise

in the k samples, use the native bayes algorithm to calculate the probability of each label and finally output

- 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: Bidirectional Encoder Representations from Transformers (BERT) was proposed by the Google AI team[9], which can be used to replace word2vec for word vector generation. The other parts of the model structure are the same as the previous model.

2.3.3 Metrics

Compared with traditional learning problems, it is very difficult to label multi-label data, and multi-label means huge classification cost[29]. In the tasks in this section, the test questions need to be labeled with multiple knowledge points. In fact, it is a multi-label classification problem. The sample dimension, data volume, and label dimension will all affect the labeling effect. This section uses the label-based metrics to evaluate the performance of the proposed test knowledge point labeling model.

Regarding multi-label classification, considering the relationship between the labels of the exercises, 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.[29], for the j-th label of the i-th example in the n examples, it can be expressed as the following formula 2.29-2.32:

$$\mathrm{TP}_j = |\{x_i | l_j \in Y_i \wedge l_j \in \tilde{Y}_i, 1 \leq i \leq n\}| \tag{2.29}$$

$$\mathrm{FP}_j = |\{x_i | l_j \notin Y_i \wedge l_j \in \tilde{Y}_i, 1 \leq i \leq n\}| \tag{2.30}$$

$$\mathrm{TN}_j = |\{x_i | l_j \notin Y_i \wedge l_j \notin \tilde{Y}_i, 1 \leq i \leq n\}| \tag{2.31}$$

$$\mathrm{FN}_j = |\{x_i | l_j \in Y_i \wedge l_j \notin \tilde{Y}_i, 1 \leq i \leq n\}| \tag{2.32}$$

where x_i and 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 2.33 and 2.35:

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{2.33}$$

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{2.34}$$

$$\mathrm{F1}_{score} = \frac{2}{\frac{1}{\mathrm{P}} + \frac{1}{\mathrm{R}}} \tag{2.35}$$

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 2.36 and 2.37.

$$\mathrm{F1}_{macro} = \frac{1}{c}\sum_{j=1}^{c}\mathrm{F1}_{score}(\mathrm{TP}_j, \mathrm{FP}_j, \mathrm{TN}_j, \mathrm{FN}_j) \tag{2.36}$$

$$\mathrm{F1}_{micro} = \mathrm{F1}_{score}(\sum_{j=1}^{c}\mathrm{TP}_j, \sum_{j=1}^{c}\mathrm{FP}_j, \sum_{j=1}^{c}\mathrm{TN}_j, \sum_{j=1}^{c}\mathrm{FN}_j) \tag{2.37}$$

Where c is the total number of labels.

Similarly, we can also count some indicators based on samples. 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 2.38-2.42.

$$\mathrm{Acc}_{ML} = \frac{1}{n}|\{i|Y_i = \tilde{Y}_i\}| \tag{2.38}$$

$$\mathrm{HmLoss} = \frac{1}{n}\sum_{i=1}^{n}\frac{\mathrm{XOR}(Y_i, \tilde{Y}_i)}{c} \tag{2.39}$$

$$\mathrm{Precision}_{\mathrm{ML}} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap \tilde{Y}_i|}{|\tilde{Y}_i|} \tag{2.40}$$

$$\mathrm{Recall}_{\mathrm{ML}} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap \tilde{Y}_i|}{|Y_i|} \tag{2.41}$$

$$\mathrm{F1}_{ML} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{2.42}$$

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. We define the set of questions involved in knowledge points j as $\mathscr{E}_j$. Remember the threshold of the number of occurrences of the knowledge point label τ^{kp}, then the knowledge points of the exercises can be divided according to the frequency of occurrence, that is, $\{j|\mathscr{E}_j| > \tau^{kp}\}$ and $\{j||\mathscr{E}_j| \leq \tau^{kp}\}$. According to the different thresholds, the ability of the model to classify knowledge points that appear frequently and that appear sparse can be tested separately.

In the experiment, five sets of thresholds of 200, 150, 100, 50, 10 are set τ_{kp}, and the exercises with the label j are recorded as $\mathscr{E}_j$, the label set that meets the requirements can be counted as $\mathscr{L}_\tau^{kp} = \{j \mid |\mathscr{E}_j| \geq \tau_{kp}\}$, the exercises containing label in $\mathscr{L}_\tau^{kp}$ is denoted as $\mathscr{E}_\tau^{kp}$, within which the average number of labels of the exercise in the set is $LAvg_\tau^{kp}$

Table 2.1 Setting of Experiment

τ^{kp}	$\lvert\mathscr{L}_{\tau^{kp}}\rvert$	$\lvert\mathscr{E}_{\tau^{kp}}\rvert$	$LAvg_{\tau^{kp}}$
200	2	463	1.21
100	22	1376	1.55
50	29	2237	1.42
10	57	3158	1.35

In this experiment, a GCN architecture is defined to learn the association between knowledge points. The architecture contains two GCN layers with dimensions of 512 and 1024 respectively. Each node uses 256-dimensional word2vec to train the label text to obtain the label embedding representation. The tag correlation threshold τ is set to multiple values to determine the model performance.

The running environment is Ubuntu 20.04, TensorFlow 2.23, Python 3.8.6, and the hardware is equipped with a Tesla V100 computing card. For each Baseline model, 5 rounds of calculation are performed, the maximum positive deviation and the minimum negative deviation are discarded and the average value is recorded.

2.3.5 Result and Analysis

The comparison experiment results are divided into the performance comparison of the horizontal Baseline model (obtained in Table 2.2-Table 2.5) and the performance comparison of different hyperparameters of the model (Table 2.4).

Table 2.2 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**

Table 2.3 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 2.4 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 2.5 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**

It can be seen from the table that the performance of the model proposed in this paper is stronger than the baseline model on the exercise training machine with a higher frequency. As the frequency of knowledge points decreases, the difficulty of classification 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 paper has achieved the best or better performance in terms of F1-Score parameters and stricter subset accuracy indicators. When the frequency of the problem labels is very low, 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 the 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 as the training set.

2.4 Summary

In the exercise recommendation system, there are a large number of exercises with missing knowledge point labels. In order to meet the requirements of the adaptive learning system, it is necessary to label the exercises, but 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 graph convolutional neural network and Bi-LSTM text mining model based on attention mechanism. After verifying the experimental data set, a better knowledge point annotation effect than existing models has been achieved.

The contributions of this chapter are as follows:

1. The relationship between knowledge points is represented by 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 over-fitting phenomenon for the multi-label classification model, resulting in performance degrada-

tion. In order to solve this problem, 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 labelled exercises can be used as the input of the knowledge tracking model to track the knowledge state of students, and can also be used as the input feature of the recommendation system to complete the recommendation of exercises based on knowledge points.

Chapter 3

Graph Attention Networks Embedded Knowledge Tracing Model With Transformer

3.1 Motivation

This section is a core part of this recommendation system, which is to obtain the student's knowledge mastery status by means of knowledge tracing. knowledge tracing is to model students' knowledge mastery based on their past answer records to obtain students' knowledge status. knowledge tracing models are abundant, and early knowledge tracing models are generally based on the Bayesian knowledge tracing (BKT)[26] of first-order Markov models, which are based on the assumption that student answers are based on a series of knowledge points that are considered to be unrelated to each other and therefore represented independently of each other, an approach that cannot capture the relationships between different concepts nor characterize complex conceptual transformations. In 2015, Piech et al. proposed the Deep knowledge tracing Model (DKT)[20], the first application of a long short-term memory network (LSTM) to the knowledge tracing task, which does not require knowledge point annotation but contains an implicit state of knowledge, achieving a baseline performance over BKT at that time, and it marked the prologue of knowledge tracing research based on neural network models. However, DKT could not output the hidden state of knowledge, and the interpretability was insufficient. Moreover, DKT used to store all memories in a hidden vector, and the prediction performance was not satisfactory for long

sequences. To address this problem, Memory Augmented Neural Network (MANN)[21] was proposed, which allows the network to keep multiple hidden state vectors and read and write these vectors separately. In 2017, Zhang et al. proposed Dynamic Key-Value Memory Networks (DKVMN)[27], which refers 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, fewer parameters, and automatic discovery of similar exercises through latent concepts. The model also has better interpretability, as it stores the problem-related potential concepts in the key matrix and the mastery of 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. To solve this problem, Sequential Key-Value Memory Networks (SKVMN)[1], 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 simply treat knowledge points as mutually independent nodes, or treat knowledge points as simple hierarchical models, but in fact, 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, we propose 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, so as 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 tasks track changes in students' knowledge status from their historical learning records, predict future learning performance, and provide 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

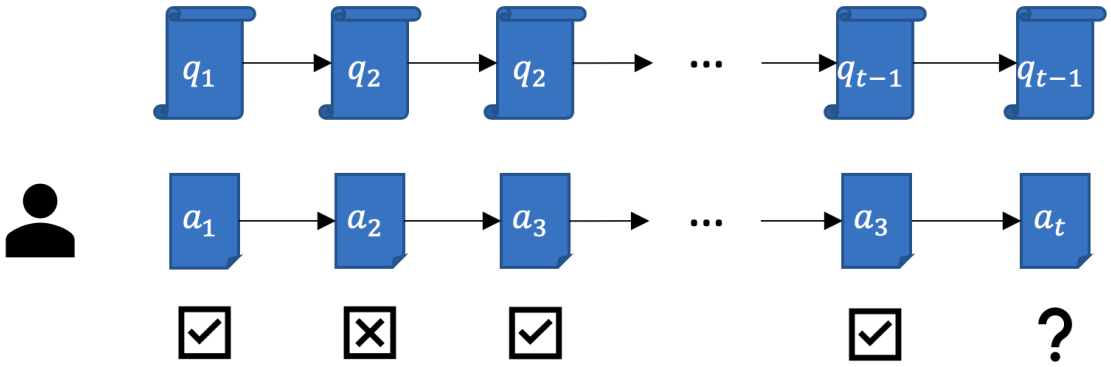


Fig. 3.1 The knowledge tracing pattern

practice is to analyze the data and process modeling of students' past answer records, so as 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 actually based on their mastery of each knowledge point. 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 basic assumption in knowledge tracing is that student performance is based on mastery of the knowledge point. The student's performance can be used to infer the student's mastery of each knowledge point, i.e., the mastery of 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 question q_t at time t, and a_t indicates the score of the question, 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 question $P(a_t = 1|a_{t-1}, \ldots, a_1)$. As shown in the figure Fig.3.1.

3.2.2 Graph Neural Networks

The traditional neural network model, which has achieved greater success in extracting features from Euclidean space data, has not worked well for applications on non-Euclidean spaces. And 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 great challenge to traditional deep learning algorithms, and the variability of the connections of the nodes of graphs leads to a huge computational effort, which makes some traditional neural network models that perform well in Euclidean space cannot be directly applied to non-Euclidean space. For example, 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 node is connected to and has dependencies on other nodes. To characterize the interdependencies between graph nodes, Graph Neural Networks (GNNs) are proposed.

There are five major classes of graph neural networks: Graph Convolution Networks (GCN), Graph Attention Networks(GAN), Graph Autoencoders(GAE), Graph Generative Networks(GGN), and Graph Spatial-temporal Networks(GSN). GCN, the pioneer of graph neural networks, simply applies the convolution operation in image processing to graph structured data processing and gives a specific derivation. It actually aggregates the features of graph neighbor nodes to make a linear transformation. To solve the problem of insufficient depth of graph propagation, multiple GCN layers can be stacked to capture the k-hop neighbor node information. However, the GCN processing requires putting the whole graph into the computational memory, which limits its computational performance. It also requires pre-inputting the graph structure information, which limits its application. To solve the problem that GCN aggregates neighbor nodes ignoring different weights of different neighbor nodes, Graph Attention Networks (GAT) introduces a masked self-attention mechanism to compute the representation of each node by assigning different weights according to the neighbor node characteristics. It does not need to use a pre-constructed graph, so only the neighbor nodes need to be known, which greatly increases the computational speed.

Graph embedding is an important research topic in graph neural networks, which represents nodes in a graph as low-dimensional vectors by preserving the network topology

and node information of the graph, which is then processed by other machine learning algorithms. In this chapter, we characterize the association between knowledge points and knowledge point information by a graph embedding algorithm to obtain the learning of embedding for knowledge structures.

3.3 Proposed Model

3.3.1 Algorithm Overview

The key point of this section is to track students' knowledge. This paper proposes a knowledge tracing model based on graph self attention network and Transformer model GATKT. In the experimental verification phase, the baseline performance of the model on several datasets achieves SOTA. The first layer of this model is the embedding aggregation layer. In this layer, a gat is used to aggregate the problem embedding and the knowledge point embedding, and to show the internal relationship between the problem and the knowledge point. It can solve the problem of data sparsity and the problem of knowledge point dependence and complex relationship. The second layer is a transformer based knowledge tracing model. Referring to the famous transformer model proposed by Google Brain [24], this layer designs a knowledge tracing model based on the transformer mechanism, which inputs questions and skills embedding, and outputs the prediction of the correct answer to the next question. Through this mechanism, the model can realize the learning of long range dependence. In addition, 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 can be seen in Fig.3.2, which is divided into the following modules:

1. GAT-based embedding layer: embedding layer based on gat: this layer uses embedding method to represent questions, knowledge points and answers. In the graph neural network, each problem represents a node, which is connected with several knowledge nodes, and these knowledge nodes are connected with the related problem nodes. Considering the relevance of knowledge points, knowledge points and knowledge points can be directly connected. The structure is shown in the Fig.3.3. This embedding layer does not carry out pre training, but carries out overall training with other layers to optimize the final result.

2. Transformer-based knowledge state encoder: this layer establishes a graph of knowledge points, which is used to track the students' mastery of knowledge points. This layer updates the knowledge state through the knowledge state, and finally it will be used 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 the problem of knowledge point dependence. Through the attention mechanism, it can also capture similar problems of knowledge points.

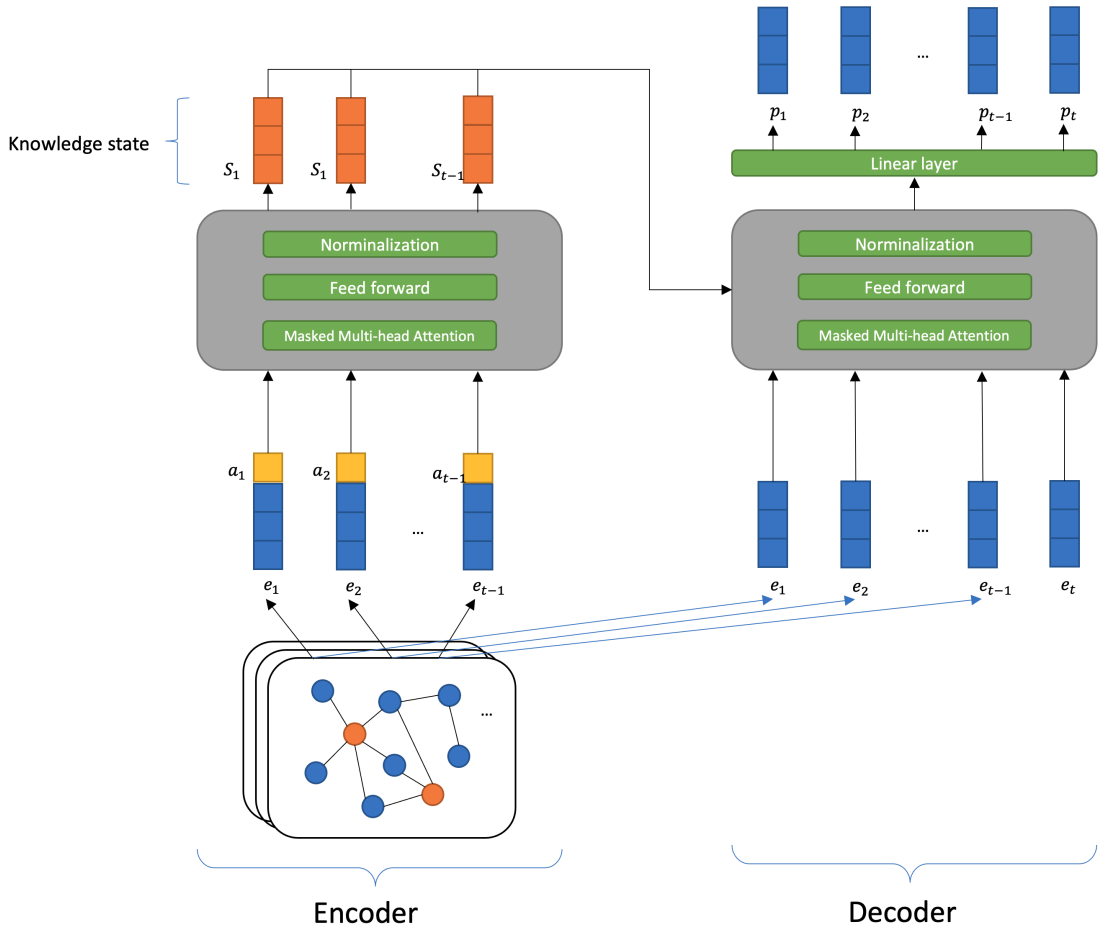


Fig. 3.2 Overview Model Structure

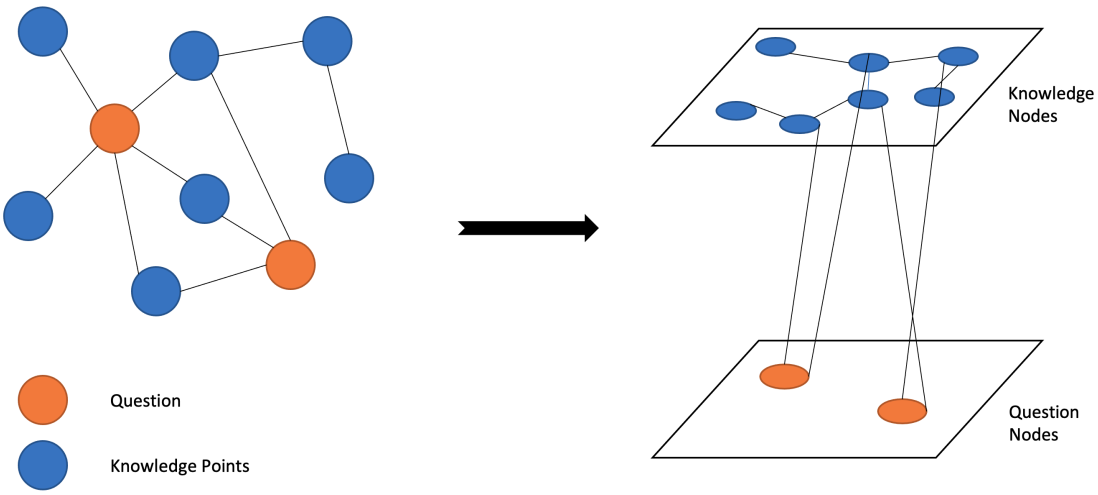


Fig. 3.3 The Graph of Knowledge Point and Question

3.3.2 GAT-based Embedding Layer

In this section, a graph self-attention network (GAT) is used to embed the problem-knowledge point association graph. It is used to reduce the data sparsity of the knowledge points in the problem data. The i-th question can be recorded as q_i, the student's answer to the i-th question 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 $X = \{x_1, x_2, ...x_{t-1}\}$, where $x_i = (q_i, a_i)$, we are based on the previous $t-1$ times To predict the probability of answering the question t correctly. For each question q_i, there are a series of knowledge points related to it. These knowledge points are denoted as $\{p_1, p_2, ..., p_{n_i}\}$. The embedding calculation here can obtain the deep-level relationship between the problems, that is, the relevance of knowledge points. This can be done ideally with a graph structure. Intuitively, you can also infer the probability of answering the question through the mastery of the knowledge points. This method can also establish a better representation of the problem-knowledge point.

In this section, the graph self attention network is used to embed the problem knowledge point association graph. The embedding method can reduce the sparsity in representing the knowledge points in question data. Let's denoted the i-th question as q_i. The answer to question I is a_i. The answer record is $x = \{x_1, x_2, \ldots, x_{T-1}\}$, where $x_i = (q_i, a_i)$. Based on the answer records of the previous $t-1$ times, we predict the correct probability of the answer to question t. For each question q_i, there are a series of knowledge points related to it, these knowledge points are denoted as $\{p_1, p_2, \ldots, p_{n_i}\}$, n_i marks the number of knowledge

points related to the question q_i. Similarly, a knowledge point p_j has also several related question $q_1, \ldots, q_{n_j}$, where n_j is the number of question related to the knowledge point p_j. So a graph $\mathbf{G}$ can be used to represent the relation of question-knowledge point, where $G = \{(q, r_{qp}, p) | q \in \mathbf{Q}, s \in \mathbf{S}\}$ and $\mathbf{Q}$ denotes question sets and $\mathbf{P}$ denotes knowledge point sets respectively. $r_{qp} = 1$ if q is related to p.

In this part, embedding calculation can obtain the deep correlation between problems, that is, the correlation of knowledge points. This can be achieved with graph structure. Intuitively speaking, we can also infer the correct answer probability of the question by mastering the knowledge points. This method can also establish a better representation of problem knowledge points. In the question-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 masked self attention mechanism. When calculating the representation of each node in the graph, it will assign different weights to its neighbors according to their different characteristics.

GAT applies an attention mechanism to weighted summation of neighboring node features. The weights of the neighboring node features depend entirely on the node features and are independent of the graph structure. In GAT, each node 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 the need to get information about the whole graph. Specifically, the node of GAT network represents the knowledge point embedding or question embedding. Denote the node as i and the neighbor node set as $\mathcal{N}_i$ and the embedding of node as x_i, the input of the graph attention layer is $x = \{\overrightarrow{x_1}, \overrightarrow{x_2}, \overrightarrow{x_3}, \ldots, \overrightarrow{x_n} | \overrightarrow{x_i} \in \mathbb{R}_F\}$ and the output is $x = \{\overrightarrow{x_1}', \overrightarrow{x_2}', \overrightarrow{x_3}', \ldots, \overrightarrow{x_n}' | \overrightarrow{x_i}' \in \mathbb{R}_{F'}\}$, where n is the number of nodes and F is the number of node features. represents the features of all nodes, and F' denotes the output node features. In order to get the corresponding input and output transformations, we need to perform at least one linear transformation based on the input features to get the output features, so we need to train a weight matrix for all nodes: $W \in \mathbb{R}^{F' \times F}$, and this weight matrix is the relationship between the F features of the input and the F' features of the output.

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

coefficients (Attention coefficients) and its softmax value are:

$$e_{ij} = a(\mathbf{W}\vec{x}_i, \mathbf{W}\vec{x}_j) \tag{3.1}$$

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

The e_{ij} and α_{ij} represents the importance of node i to node j.

In General, the complete attention mechanism is:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLu}(\overrightarrow{\text{a}}^T[W\vec{h}_i \| W\vec{h}_j]))}{\sum_{k\in N_i}\exp(\text{LeakyReLu}(\overrightarrow{\text{a}}^T[W\vec{h}_i \| W\vec{h}_k]))} \tag{3.3}$$

where $||$ means concatenation operation. It is a single-layer feedforward neural network, $\overrightarrow{\text{a}} \in \mathbb{R}^{2F'}$ is the weight matrix connecting the layers in the neural network, and a LeakyReLu function is also added to the output layer of this feedforward neural network. The procedure is like Fig.3.4;

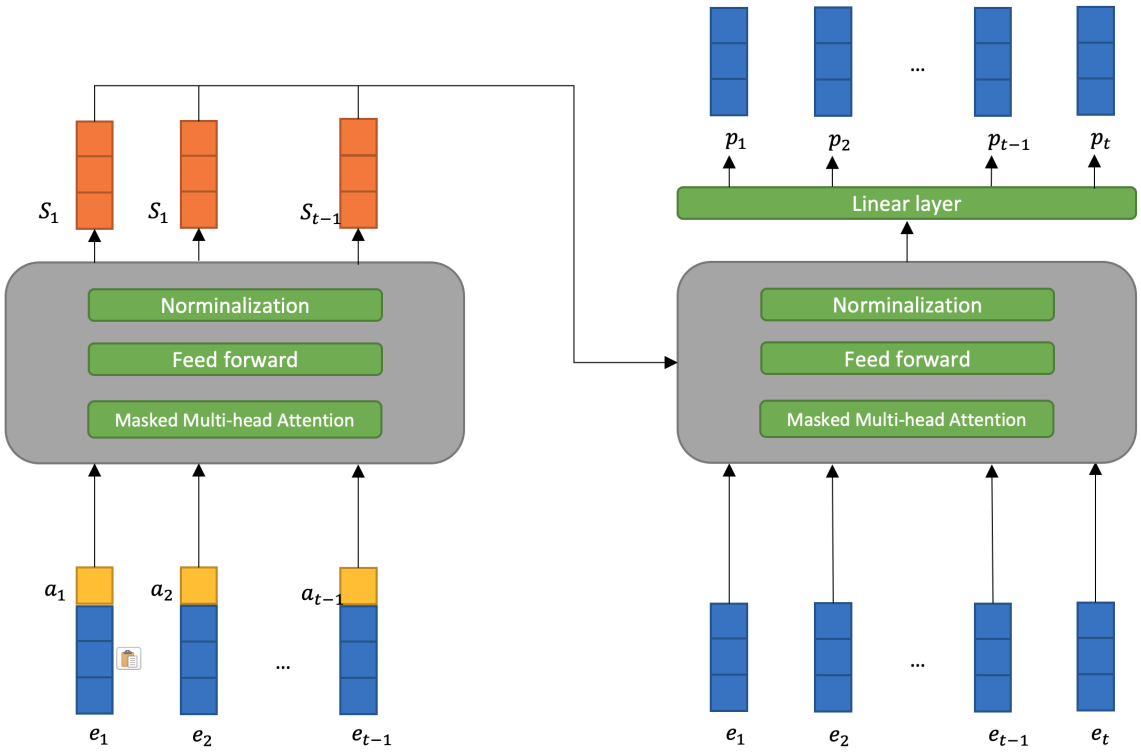


Fig. 3.4 Feature extraction and attention mechanism

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 node then gets a weight about the neighboring nodes and then the combined representation of the nodes is obtained by summing.

$$\vec{x}_i' = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{x}_j) \tag{3.4}$$

where σ are the non-linear transformation such as ReLU or LeakyReLU. If the edge $j \to i$ does not exist, we can simply omit the calculation of α_{ij}, which can reduce the calculation. The formula means that the output characteristics of this node are related to all the nodes adjacent to it and are obtained after the nonlinear activation of their linear sum.

To make the result of self-attention more stable. The multi-headed attention mechanism can be used here. Multi-head attention is actually a combination of multiple self-attention structures, each head learns features in a different representation space, and multiple heads may learn slightly different attention focus, which gives the model more capacity. K independent attention mechanisms can be achieved by connecting k features together.

$$\vec{x}_i' = \|_{k=1}^{K} \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{x}_j) \tag{3.5}$$

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 be used to predict the output characteristics of each node:

$$\vec{x}_i' = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{x}_j) \tag{3.6}$$

There are a total of K attention mechanisms to consider, with k denoting the k-th of K.

From this layer, we got the embedded question and knowledge point vector $\hat{q}_i = \vec{x}_i'$, which would be passed into the Transformer Block in Knowledge tracing module.

3.3.3 Transformer-based knowledge state encoder

The input of transformer Block is the concatenation of the embedded question $\hat{q}_i$ and corresponding answer a_i, i.e. $e_i = \hat{q}_i \| a_i$. The general structure is like Fig.3.5. The traditional Transformer model is modified to apply to knowledge tracing task. The Transformer can learn the W-matrix relating underlying knowledge points and questions rather than directly learn the representation of each question. 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.

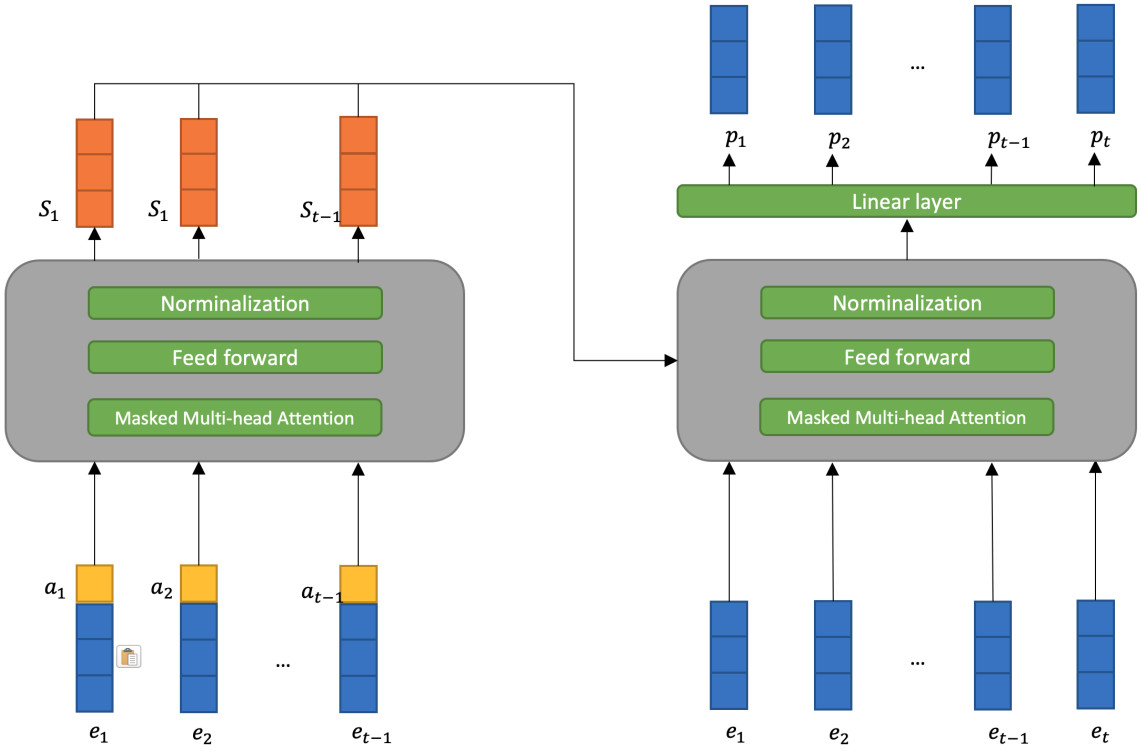


Fig. 3.5 The Transformer architecture

The output of Embedding layer e_i is passed into a Transformer Block, which is a self-attention module.

$$q_i = Qe_i, k_i = Ke_i, v_i = Ve_i \tag{3.7}$$

$$A_{ij} = \frac{q_i k_j + b(\Delta t_{i-j})}{\sqrt{d_k}}, \forall j \leq i \tag{3.8}$$

$$h_i = \sum_{j \leq i} \text{softmax}(A_{ij}) v_j \tag{3.9}$$

the masked attention layer first extracts query gi, key k;, and value y; from the inputs ei. It then assign an attention, Aii, to a past interaction e, based on two components:

1. $q_i \cdot k_j$, the query-key agreement between e_i and e_j, which could be interpreted as the degree of latent skills overlapping between interaction e_i and e_j;

2. A time gap bias, $b(\delta t_{i-j})$, which adjusts the attention weight by the time gap between interactions e_i and e_j. The hidden representation h_i is a weighted sum of the past value representations of e_i.

A Transformer block also have feedforward layer, normalization layer, and residual connections. The outputs of a stack of Transformer blocks is feed to the linear layer before calculating the final loss. e_{i+1} and e^*_{i+1} are the results of applying Interaction Mapping Layer to $(\hat{q}_{i+1}, 1)$ and $(\hat{q}_{i+1}, 0)$.

$$p_{i+1} = \frac{\exp(h_i e_{i+1})}{\exp(h_i e_{i+1}) + \exp(h_i e^*_{i+1})} \tag{3.10}$$

$$\text{Loss} = -\sum_i c_{i+1} \log(p_{i+1}) + (1 - c_{i+1}) \log(1 - p_{i+1}) \tag{3.11}$$

3.4 Experiments

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

3.4.1 Datasets

After investigating the existing knowledge tracing model, we 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 questions 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 questions, 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 questions, 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 questions, 156 knowledge tags and about 189 thousand interactions.

Table 3.1 Dataset Statistics

Dataset	#Students	#Questions	#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 tracking 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 tracking, 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 this approach can be adopted.

For each dataset, 70% is used as the training set, and the remaining 30% is used 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 tracking models proposed recently. The following models participate in performance comparison.

- Bayesian knowledge tracing (BKT)[26]: BKT is a classic model of knowledge tracking. It is based on Hidden Markov Model (HMM), which models the knowledge state of learners as a set of binary variables corresponding to the mastery of knowledge points.

- DKT is the first deep learning-based knowledge tracking model, which 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)[27]: 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, fewer parameters, and automatic discovery of similar exercises through latent concepts.

- Graph-Based Knowledge Tracing (GKT)[19]: GKT applies graphical neural networks to knowledge tracking 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 experimental results are shown in the Table 3.2, and the proposed model achieves the best performance metrics compared to other 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, while 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, while the performance difference between DKT, DKVMN, and GKT is smaller, due to the smaller association between knowledge points, and thus the performance improvement from considering the association between knowledge points is smaller. 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 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 tracking model based on graph self-attentive network hungry mbedding 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 output, which can be used in the subsequent exercise recommendation process. At the same time, the

knowledge state sequence and the exercises are used as the input of the subsequent decoder 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 graph relational network with separation of exercises and knowledge points is 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 to better model the change of students' knowledge states.

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 reasonable tuning of parameters.

Chapter 4

Exercise Recommendation System Based on Knowledge State

4.1 Motivation

High school mathematics intelligent question bank is built by learning from the model of adaptive learning. Adaptive learning is the theoretical abstraction of adaptive learning system model. Many adaptive learning system models focus on the construction of domain model, student model and adaptive engine. This paper proposes an intelligent recommendation algorithm based on individual characteristics, which integrates domain model, student model and adaptive engine The self-adaptive engine abstracts the high school mathematics question bank into the characteristic question bank according to the domain model, and abstracts the student model into the individual characteristic bank. The self-adaptive engine is based on the individual characteristic bank through the intelligent algorithm, dynamically selects the training questions suitable for the individual from the characteristic question bank, and realizes the personalized recommendation of the question bank.

Traditional paper teaching materials are usually printed in large quantities. Due to the limitation of paper space and in order to adapt to the majority of students, one or two representative test questions have to be selected according to each chapter. In order to meet the comprehensiveness of knowledge points and the public adaptability, such teaching aids lead to the small scale of test questions, the incomplete coverage of questions and the failure to meet the requirements of students' targeted training. E-learning has brought a reform in the field of teaching with its flexible learning style, avoiding large class teaching, and

students can choose their own learning resources through autonomous navigation. Online question bank with eleaning As the times require, the early online question bank simply moves the offline resources to the Internet. Although students can choose their own topics and find the test questions they want for intensive training, it is lack of interactivity. Students do not get timely and effective feedback after finishing the questions, and can not accurately locate the strengths and weaknesses of students, which is very limited in improving students' performance. In view of these shortcomings, it is the current research direction of intelligent question bank to build a student-centered system, in which students can feed back their mastery of knowledge points in real time by doing questions, and build a complete user portrait of students. The question bank can intelligently diagnose, screen and push targeted questions according to the submission of students' questions.

Adaptive learning means that learners choose different learning styles depending on the content they are learning, and that they continue to discover their own "personalized" learning style as they learn. Adaptive learning is not a simple transfer of offline resources to online, nor is it a simple combination of "Internet + education". Rather, it focuses on the individual student and provides different personalized learning programs for people with different cognitive levels and styles.

The concept of adaptive learning has been a hot topic of research in the education field and has been one of the theories studied by many educators. Since Brusilovsky proposed a general model of adaptive learning system, the concept of adaptive learning was first applied to engineering practice, but there has been a lack of standard reference models and architectural systems until the first adaptive hypermedia application model was developed, and later there were LAOS, XML adaptive media model and WebXML reference model. Since then, adaptive learning has been transformed from theoretical research in education to application development in computing, and many adaptive learning models have focused on the construction of domain models, student models, and adaptive engines.

The domain model focuses on building student learning resources, which are not just student textbooks or tutorials, but also structured entities in the domain knowledge, and ontologies or semantic webs are used to build connections between entities to form a complete knowledge base, including fine-grained knowledge concepts and their attributes, examples, exercises, materials, and even graphical images and audio.

The student model is mainly used to build a model with individual student characteristics, which can be described as individual students' mastery of each test topic in the domain

model, and the student model will change with the changes of entity relationships in the domain model. In addition, the student model is also described in CELTS-11 according to our educational technology specification on the learner specification model, which adds additional information such as the student's age, background, and region.

The adaptive engine, also known as the instructional selection strategy, uses adaptive interpretation rules from the domain model to the student model to enable the system to select the domain model that meets the student's requirements based on the student's personality characteristics, either through rules or algorithms. Currently, it is popular to use big data technology to recommend students' personalized learning solutions by using popular recommendation algorithms such as collaborative filtering recommendation, content recommendation, and hybrid recommendation through statistics of students' learning paths or habits.

In this paper, we further optimize the domain model, student model, and teaching strategy in the adaptive reference model, and apply the optimized model to build an intelligent question bank system. The feature database is structured according to the dimensions of forgetfulness, abstractness, indirectness, comprehensiveness, and computational complexity, and individual students are trained to generate a database of individual features reflecting their knowledge acquisition ability in real time. The intelligent question selection algorithm is used to dynamically select the appropriate training set from the question database based on the individual feature database.

4.2 Proposed Model

4.2.1 Algorithm Overview

This chapter proposes an adaptive recommendation method for test questions that combines graph neural network knowledge tracking and collaborative filtering. The method models students' knowledge states through the GAT-based knowledge tracking model model designed in the previous chapter, and uses the clustering method of graph neural networks to perform knowledge-based clustering of test questions so that a series of relevant exercises can be obtained. The first part is the recall part, i.e., candidate test generation, which generates a series of similar exercises, and this is done by clustering the test questions through graph neural networks, which apply graph clustering methods to cluster the test questions based on knowledge points in a reasonable way. The second part is the ranking part, which

combines the knowledge state of students obtained from the knowledge tracking system with co-filtering.

1. Exercise candidate generation: The role of this part is to generate a series of exercises that match the student's current learning state, and quickly filter out a preliminary set of exercises, and the subsequent part will generate the most suitable N exercises from this set and sort them by collaborative filtering algorithms. It designs a multi-layer MLP network with reference to the youtube recommendation system. It concatenates the student learning state embedding vector generated by knowledge tracking with the student's problem record embedding, personalized information, etc. as input. After a series of ReLU networks and then output a probability distribution over all candidate exercises by softmax normalization.

2. Ranking: This part is to generate Top N recommendation exercise. It implements the ranking of the recommended test questions using a neural network architecture similar to that of the previous section. Its inputs are the user's knowledge structure, recent exercises, and common practice questions from similar users.

The structure of the recommendation system is like Fig.4.1

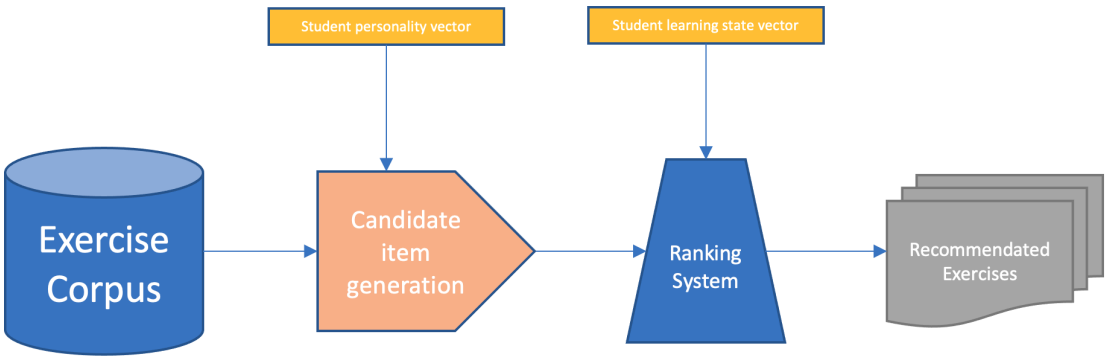


Fig. 4.1 The Architecture of Recommendation System

4.2.2 Recall Stage

In the recall phase, due to the large amount of data, the algorithm of Collaborative Filtering (CF) can achieve half the result with twice the effort. It only needs to calculate the

similarity of students' knowledge states h' with other students who have similar knowledge states h.

Collaborative Filtering, the most classic type of recommendation algorithm, includes both online collaborative and offline filtering. The so-called online collaborative is to find items that users may like through online data, while offline filtering is to filter out data that are not worth recommending, such as data with low recommendation value ratings, or data that users have already purchased despite high recommendation values. The model of collaborative filtering is generally m recommended items, m users' data, only some users and some data are rated between the data, other parts of the rating is blank, at this time we want to use the existing part of the sparse data to predict the rating relationship between those blank items and data to find the highest rated items recommended to users.

Generally speaking, there are three types of collaborative filtering recommendations. The first one is user-based collaborative filtering, the second one is item-based collaborative filtering, and the third one is model-based collaborative filtering.

Student Similarity Calculation

The purpose of collaborative filtering is to recommend recommended items of users similar to the target user to the user. The user-based collaborative filtering algorithm is discovered through historical behavior data of users, and measures and scores these preferences. Set the similarity of users through a given algorithm, and make recommendations among users who have the same preferences.

User-based collaborative filtering mainly considers the similarity between users and users. As long as we find out the items that similar users like and predict the target users' ratings of the corresponding items, we can find the items with the highest ratings and recommend them to users. The item-based collaborative filtering is similar to the user-based collaborative filtering, except that we turn to find the similarity between items and users, and only if we find the ratings of certain items by target users, then we can predict similar items with high similarity and recommend the highest rated similar items to users. For example, if you buy a machine learning related book online, the website will immediately recommend a bunch of machine learning, big data related books to you, and the idea of collaborative filtering based on items is obviously used here.

Similar statistics are used to get neighboring users with similar hobbies or interests, so we can use the obtained user's knowledge state embedding h_t from the previous knowledge

tracking module, which can be used as the basis for calculating the similarity. The first step is to find other users who are similar to the new user based on their historical behavior information; at the same time, to predict the items that the current new user may like based on the evaluation information of these similar users on other items. Given the user rating data matrix R, the user-based collaborative filtering algorithm needs to define the similarity function $s : U \times U \rightarrow R$ to calculate the similarity between users, and then calculate the recommendation results based on the rating data and the similarity matrix. We can use the cosine similarity to calculate this value.

$$s(u,v) = \frac{h_u \cdot h_v}{\|h_u\|_2 \|h_v\|_2} \tag{4.1}$$

where h_u and h_v represent the knowledge mastery state of user u and v. We can obtain the exercise recommendation history $\mathscr{H}$ of user B, which is closest to user A to be recommended, and then use the exercises in $\mathscr{H}$ as a rough set as the next sorted list of exercises.

Exercise Filtering System

Through the previous section, we calculated the similarity of students, which comprehensively considers students' current knowledge proficiency, students' personalized information and so on. Next, we need to use the collaborative filtering algorithm to generate a rough recommended set of exercises S_{raw}.

When the recommendation system is running, the system will record the students' question records. After calculating the similarity of the students, they can be sorted according to the similarity. Given a threshold, list the students whose similarity is greater than the threshold, and divide the students according to the list. The exercises corresponding to the problem record are added to S_{raw}.

The algorithm is as 1:

4.2.3 Ranking Stage

In the previous step we obtained a list of recommended exercises for similar students by collaborative filtering in the recall phase, and in this phase, a ranking of the exercises is needed to further reduce the amount of recommendations and give the ranking sequence of exercise. In this section, we design an MLP-based test ordering model. The model

Algorithm 1 Exercise Filtering Algorithm

Require:
The target student s_i;
The student represent matrix, $S = \{s_0, s_1, \ldots, s_N\}$;
The log of recommendation $L = \{L_0, L_1, \ldots, L_N\}$
The log-exercise relation: $L_i = \{E_0, \ldots, E_{|L_i|}\}$
The filtering thread T;
Ensure:
The filted exercise set $S = \{E_{s_0}, \ldots, E_{s_N}\}$
Calculate the similarity between s_i and other students *Sim*
Sort *Sim* and get the top T results $R = \{s_{r_0}, \ldots, s_{r_T}\}$;
Aggregate the exercises log of students in R, get exercise set S
return S;

inputs the student's knowledge state vector h_t, the knowledge state change vector d_h learned through sequence embedding, and the knowledge point labeling vector of the exercises. The architecture is like Fig.4.2.

The model is a three-layer multi-layer perceptron, with two fully connected layers in the middle, and the output layer uses softmax to output the probabilities of exercise recommendation. The input layer inputs the student's knowledge state vector, knowledge state change vector and exercise knowledge point label vector. The model is a three-layer multi-layer perceptron, with two fully connected layers in the middle, and the output layer uses softmax to output the probabilities of exercise recommendation. The input layer inputs the student's knowledge state vector, knowledge state change vector and exercise knowledge point label vector. The loss function is:

$$\mathscr{L} = \sum_{i=1}^{T} y_i \log(\text{sigmoid}(\hat{y}_i)) + (1 - y_i) \log(1 - \text{sigmoid}(\hat{y}_i)) \tag{4.2}$$

where $\hat{y}_i$ is the output value, y_i is actual recommended weights for the exercises.

After training the model, sort the output values, and the sorted list is the recommended problem set.

4.3 Summary

This section proposes a two-stage recommendation system framework based on recall-ranking. In the recall stage, collaborative filtering is used, and in the ranking stage, multi-layer

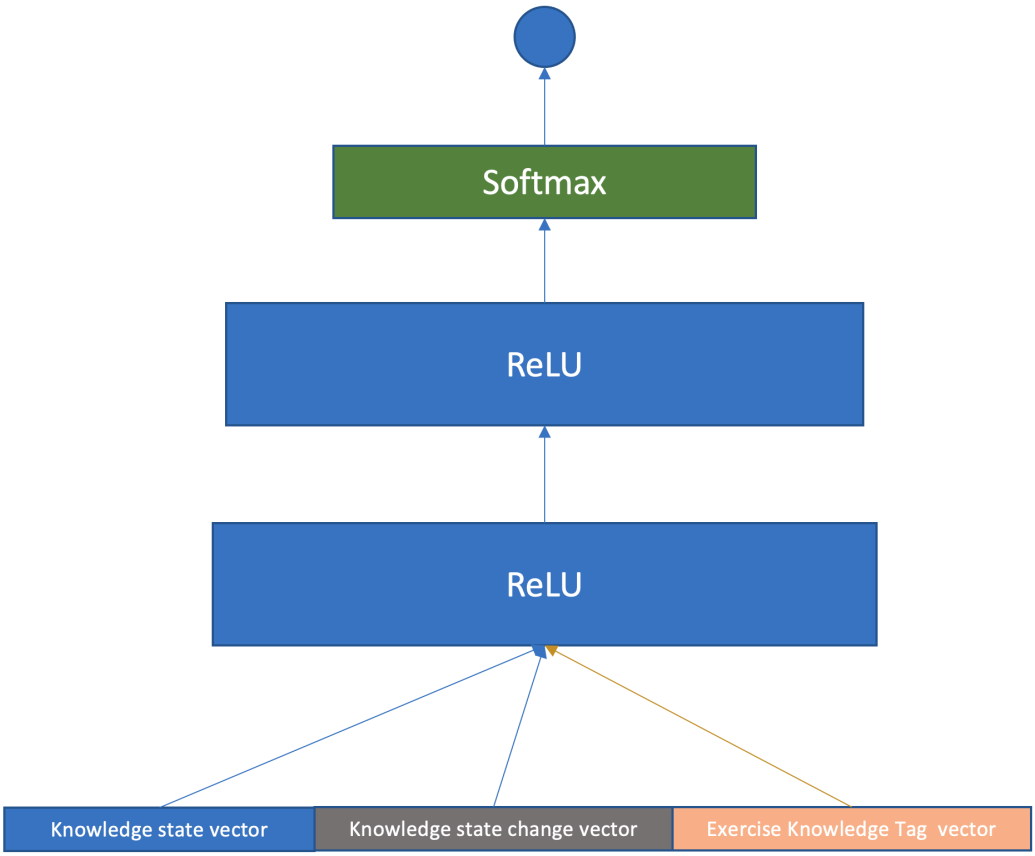


Fig. 4.2 The Architecture of Recommendation System

perceptron are used to model students' knowledge state-exercise knowledge points. At the same time, taking into account the student's learning speed, that is, the change of knowledge state, the output exercises match the current knowledge state.

Chapter 5

Conclusion and Future Work

In this paper, a test question recommendation system based on graph neural network combined with knowledge mapping and factorization algorithm is established, which generally accomplishes the design goals and is original in the three stages of the recommendation system-data source generation, knowledge tracking and knowledge recommendation, etc. This paper adopts the current more popular graph neural network model and makes some improvements to the existing model to solve the tasks of knowledge analysis of test questions and knowledge state tracking of students, and achieves the desired results in the experiments with some performance improvements to the existing model.

Among them, in the data source part, they are input to the initial database by crawlers and manual input, and then the automatic knowledge point analysis model of test questions based on text mining and graph neural network iterative learning is used to complete the knowledge point label mining of test questions, and they then establish the knowledge map of high school mathematics by some means of knowledge map construction, which solves the "zero resource " problem. In terms of the knowledge tracking of students, this paper innovatively applies the graph self-attentive network to this task, and through the improvement of the model and the consideration of several factors, the task has been improved somewhat compared with the existing model and has improved in performance. Finally, by labeling the knowledge of the test questions and tracking the students' knowledge, the factorization machine algorithm is used to complete the learning resource recommendation, which solves the cold start problem and accomplishes the design goal of adaptive learning.

In future work, the sequential nature of the model can be solved by combining other graph neural network models or adding some memory mechanisms.

References

[1] Abdelrahman, G. and Wang, Q. (2019). Knowledge tracing with sequential key-value memory networks. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[2] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[3] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

[4] Chaudhry, R., Singh, H., Dogga, P., and Saini, S. K. (2018). Modeling hint-taking behavior and knowledge state of students with multi-task learning. *International Educational Data Mining Society*.

[5] Chen, Z.-M., Wei, X.-S., Wang, P., and Guo, Y. (2019). Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5177–5186.

[6] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*.

[7] Cui, W., Xue, Z., and Thai, K.-P. (2018). Performance comparison of an ai-based adaptive learning system in china. In *2018 Chinese Automation Congress (CAC)*, pages 3170–3175. IEEE.

[8] Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. (2018). Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[9] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[10] Fu, J., Zheng, H., and Mei, T. (2017). 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.

[11] Guo, W., Wang, J., and Wang, S. (2019). Deep multimodal representation learning: A survey. *IEEE Access*, 7:63373–63394.

[12] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

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

[14] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[15] Liu, W., Shen, X., Wang, H., and Tsang, I. W. (2020). The emerging trends of multi-label learning.

[16] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space.

[17] Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J., and Khudanpur, S. (2011). Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5528–5531. IEEE.

[18] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

[19] Nakagawa, H., Iwasawa, Y., and Matsuo, Y. (2019). 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.

[20] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., and Sohl-Dickstein, J. (2015). Deep knowledge tracing. *Advances in neural information processing systems*, 28:505–513.

[21] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR.

[22] Sun, M., Yuan, Y., Zhou, F., and Ding, E. (2018). Multi-attention multi-class constraint for fine-grained image recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 805–821.

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

[24] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

[25] Wu, L., Sun, P., Hong, R., Fu, Y., Wang, X., and Wang, M. (2018). Socialgcn: an efficient graph convolutional network based model for social recommendation. *arXiv preprint arXiv:1811.02815*.

[26] Yudelson, M. V., Koedinger, K. R., and Gordon, G. J. (2013). Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*, pages 171–180. Springer.

[27] Zhang, J., Shi, X., King, I., and Yeung, D.-Y. (2017). Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*, pages 765–774.

[28] Zhang, M.-L. and Zhou, Z.-H. (2007). Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048.

[29] Zhang, M.-L. and Zhou, Z.-H. (2013). A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

[30] Zhou, P., Shi, W., Tian, Qi, Z., Li, B., Hao, H., and Xu, B. (2016). 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.