SQL-DP:A Novel Difficulty Prediction Framework for SQL Programming Problems

Jia Xu*
Guangxi University
Guangxi, China
xujia@gxu.edu.cn

Tingting Wei
Guangxi University
Guangxi, China
tingtingwei@st.gxu.edu.cn

Pin Lv†
Guangxi University
Guangxi, China
lvpin@gxu.edu.cn

ABSTRACT

In an Intelligent Tutoring System (ITS), problem (or question) difficulty is one of the most critical parameters, directly impacting problem design, test paper organization, result analysis, and even the fairness guarantee. However, it is very difficult to evaluate the problem difficulty by organized pre-tests or by expertise, because these solutions are labor-intensive, time-consuming, leakage-prone, or subjective in some way. Thus, it is of importance to automatically evaluate problem difficulty via information technology. To this end, we propose a novel difficulty prediction framework, named SQL-DP, for Structured Query Language (SQL) programming problems, mastering which plays a vital role in learning the database technology. In SQL-DP, semantic features of problem stems and structure features of problem answers in the form of SQL codes are both computed at first, using the NLP and the neural network techniques. Then, these features are used as the input to train a difficulty prediction model with the statistic error rates in tests as the training labels, where the whole modeling does not introduce any experts, some as knowledge labeling. Finally, with the trained model, we can automatically predict the difficulty of each SQL programming problem. Moreover, SQL programming problem answering log data of hundreds of undergraduates from Guangxi University of China are collected, and the experiments conducted on the collected log data demonstrate the propped SQL-DP framework outperforms the state-of-the-art solutions apparently. In particular, SQL-DP decreases the RMSE of difficulty prediction by at most 7.23%, compared with the best-related framework.

Keywords

Problem difficulty prediction, SQL programming problem, Code structure, Neural network, Intelligent tutoring system

1. INTRODUCTION

*Corresponding author.
†Corresponding author.

J. Xu, T. Wei, and P. Lv. SQL-DP: A novel difficulty prediction framework for SQL programming problems. In A. Mitrovic and N. Bosch, editors, *Proceedings of the 15th International Conference on Educational Data Mining*, pages 86–97, Durham, United Kingdom, July 2022. International Educational Data Mining Society.

https://doi.org/10.5281/zenodo.6852986

Stem:	Query the name of students who have taken the course with the course id 'C1'.
Answer:	SELECT Sname FROM Student WHERE Sno IN (SELECT Sno FROM SC WHERE Cno = 'C1');
Difficulty:	0.4

Figure 1: An example of SQL programming problem.

With the progress of information technology, Intelligent Tutoring System (ITS) services are broadly applied, where problem (or question) difficulty has become one of the most critical parameters. The problem difficulty refers to the percentage of students who wrongly answer the problem [14]. Given the information of problem difficulty, an ITS can recommend exercises of suitable difficulty to students with varied knowledge proficiency [31], can automatically organize a test paper by choosing questions with different difficulty levels [12], and can better achieve a fairness guarantee for various other types of education tasks [29]. However, the difficulty of a problem is not directly observable before the test is conducted. To predict the problem difficulty in advance, traditional methods often resort to expertise (e.g., experienced teachers) who are asked to manually label the question difficulty according to their experience, or artificial tests organization [15]. Unfortunately, these human-based solutions are limited in that they are labor-intensive, time-consuming, leakage-prone, or subjective in some way [16]. Therefore, there is an urgent need to design problem difficulty prediction methods without manual intervention.

Recently, several non-human-based solutions that rely on machine learning techniques have been proposed [16, 23, 29, 18, 21, 8, 11]. For example, in [16], Huang et al. present TACNN, a test-aware attention-based convolutional neural network to automatically solve the difficulty prediction task for reading comprehension problems in standard English tests. In specific, TACNN utilizes the information of reading passage, question, option, and answer together to predict the difficulty. As another example, Qiu et al. in [23] propose a document enhanced attention-based neural Network (DAN) to predict the difficulty of multiple-choice problems in medical exams. Besides considering stem, option, and answer, DAN fetches relevant medical documents to enrich the information of each question. Moreover, in [29], Tong et al. design a group of salable data-driven models, i.e., C-MIDP and R-MIDP, based on CNN and RNN neural network architectures to predict the difficulty of mathematical

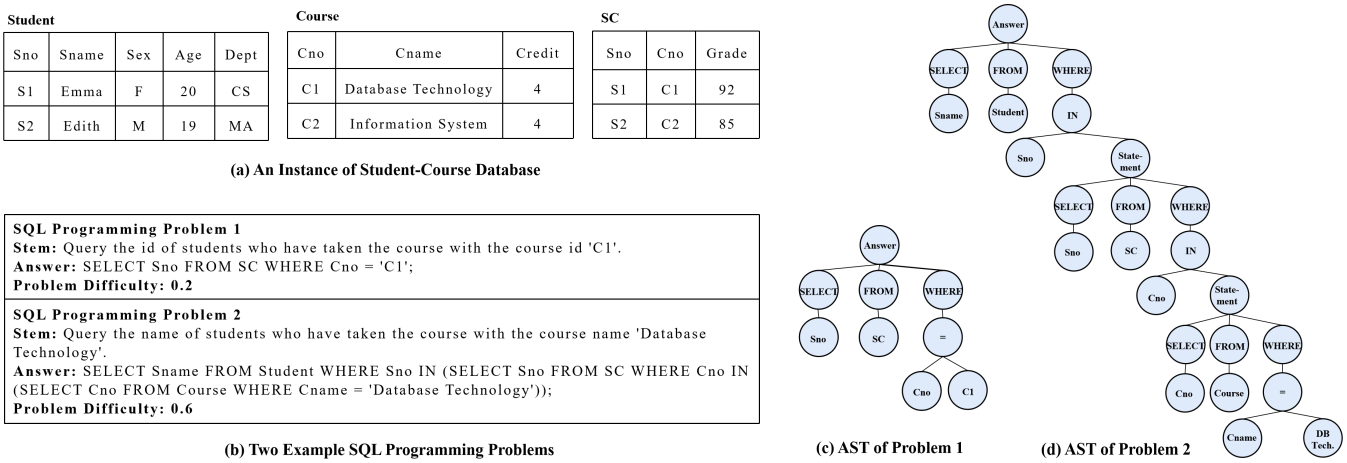


Figure 2: Examples of SQL programming problems and different code structures of their answers.

questions, which are leveraged by the historical test logs and the corresponding item materials (e.g., stem, option). However, no work is still proposed to estimate the difficulty of Structured Query Language (SQL) programming problems. Structured Query Language (SQL) is the de-facto database query language widely used in industry and taught in almost all computer-related majors in universities [19, 28], we thus focus on solving the difficulty prediction for SQL programming problems in this paper. Figure 1 shows a SQL programming problem containing a stem, an answer, and a difficulty label.

Being different from the existing non-human based solutions which only emphasizes the analysis of stem and option of problems, the difficulty of a SQL programming problem is not only influenced by the stem, options, or answer of the problem, but also depends to a great extent upon the structure of the SQL code answer to the problem. Taking the two SQL programming problems defined over the 'Student-Course Database' in Figure 2 as an example. Note that the 'Student-Course Database' has three tables, namely Student, Course, and SC, which records student-course selection relationships. Though the two examples of SQL programming problems in the figure have very similar stems, their standard answers in the form of SQL codes exhibit significantly different code structures depicted by Abstract Syntax Trees (ASTs) [17], which makes their difficulty values apparently different from each other.

To this end, we propose SQL-DP, a novel difficulty prediction framework for SQL programming problems and name it as SQL-DP. SQL-DP consists of two major modules. First, the feature extraction module is responsible for computing representations of SQL programming problems by not only considering the semantic information in their stems, but also taking the structure information of their answers in the form of SQL code into account. Using the computed representations of SQL programming problems as the input, then the difficulty prediction module in SQL-DP trains a difficulty prediction model with the statistic error rates in tests as the training labels. The trained difficulty prediction model can be applied to estimate the difficulty of a SQL programming problem in an automatic manner. Note that the whole modeling in SQL-DP does not introduce any human intervention or knowledge labeling, ensuring the usability of the solution in a wide range of application scenarios. The Main contributions of this paper include:

- We propose SQL-DP, a novel difficulty prediction framework for SQL programming problems, which not only considers the semantics of problem stems, but also leverages the information of code structures of problem answers to quantify the difficulty of the problem. As far as we know, this is the first systematic solution to predict the difficulty of SQL programming problems.
- We collect the stems, answers, and answering results of hundreds of SQL programming problems by organizing undergraduates to complete tests of SQL programming problems using our self-developed SQL online judge (OJ) system.
- We conduct a group of experiments using the collected physical-world dataset, and the experimental results show the superiority of the proposed SQL-DP framework in predicting the difficulty values of SQL programming problems.

The rest of our paper is organized as follows. Section 2 discusses the related works; Section 3 states the preliminaries of this work; Section 4 describes details of the proposed SQL-DP framework for the prediction of SQL programming problems; then Section 5 evaluates the effectiveness of the proposed SQL-DP framework using physical-world dataset; finally Section 6 concludes the paper and points out future research directions.

2. RELATED WORKS

Problem (or question) difficulty prediction is an important problem having been widely studied in educational domain.

Traditionally, the difficulty of a question is predicted and labeled by expertise (e.g., experienced teachers) according

to their experience[1], which is labor-intensive and subjective, and not applicable when there are too many questions. An alternative solution is to organize an artificial test (also called as pretest) [1, 15], where a small group of students are required to take part in the artificial test and their error rates are then used to estimate the question difficulty. To improve the estimation precision, a group of educational measurement theories are applied, including *Classical Test Theory* (CTT) [6], *Item Response Theory* (IRT) [9], and *Cognitive Diagnosis Theory* [30]. Many well-known tests, such as Test of English as Foreign Language (TOEFL) and Scholastic Assessment Test (SAT) predict question difficulty following such solution. Unfortunately, the artificial tests based difficulty prediction solution is limited in that it is not only labour-intensive, time-consuming, but also has the risk of question leakage [16].

To overcome the shortcomings of traditional methods, with the development of machine learning technologies, many non-human based solutions of problem difficulty prediction emerge, which can be divided into two categories, i.e., simple regression analysis based and artificial neural network based.

Simple regression analysis based methods establish a simple regression model (e.g., linear regression, multiple regression, and SVM) to construct a mapping function between question difficulty and its influence factors, and the difficulty of new questions can be predicted based on the regression model. As an early example, Chon and Shin [3] built a difficulty prediction model for multiple-choice reading test items by using the multiple regression technique and maximum likelihood estimation. Several features of items, such as response time of testees and paragraph length, are set as the influence factors of item difficulty. A difficulty estimation model based on correlation and regression analysis for English vocabulary questions was then discussed in [27]. As another example, in [25], Makoto Sano proposed to extract a series of language features from multiple-choice questions of reading comprehension, and analyze the extracted features using multiple regression models to obtain the features most related to the question difficulty. Moreover, in [8], Masri et al, proposed to analyze influence factors (e.g., topic and depth of knowledge) of difficulty questions in the sixth grade science test of British primary schools, and establishes a stepwise regression model to predict the difficulty of questions. A difficulty prediction model was also presented towards suggestive blank filling questions for English Tenses [21], which employs the ridge regression model to analyze many factors (e.g., questions text and blank filling words) affecting the question difficulty. These regression analysis based methods, however, have limitations, since they require some domain knowledge and artificially define the factors affecting the question difficulty.

In view of the limitations of simple regression analysis, many researchers proposed to learn the complex relations between influencing factors and question difficulty via artificial neural networks, and as a result achieved the goal of automatic question difficulty prediction. For example, in [16], Huang et al. proposed a model, named TACNN, to predict the difficulty of reading comprehension questions in English test via Convolutional Neural Network (CNN), by using the information of reading passage, question, options, and answer of each question. Similarly, Tong et al. in [29] proposed a prediction method for the difficulty of mathematical questions based on both of CNN and Recurrent Neural Network (RNN), by analyzing stem, options, and answers of every question. Besides, in [11], Hsu et al. introduced a novel method for automated estimation of multiple-choice items which consist of the following item elements: a question and alternative options. The proposed method utilizes neural network to learn embeddings of question materials in semantic space. Then, it computes the semantic similarities among the stem, answer, and distractors, which are then together fed to a SVM for training the question difficulty prediction model. Then, in [18], Lin et al. proposed a question difficulty prediction model for Chinese reading comprehension problems based on long-term and short-term memory artificial neural network, while in [23] proposed a difficulty prediction method, named as DAN, for multiple-choice questions in medical examination based on neural network model. In specific, besides considering stem, options, and answer, DAN fetches relevant medical documents to enrich the information of each question.

Although recent years have witnessed many works that predict problem difficulty automatically without manual intervention. However, none of these works focus on solving the difficulty prediction of SQL programming problems whose answers in the form of SQL codes are significantly different from their answers. Considering the structure information of SQL codes has great impact on the difficulty of SQL programming problems, all existing works hence can not well handle the difficulty prediction problem for SQL programming items which is discussed in this paper.

3. PRELIMINARY

In this section, we first introduce the method of capturing code structure information. Then we formally define the problem of difficulty prediction for SQL programming problems.

3.1 Code Structure Extraction

In [13], Hindle et al. demonstrate that programming languages, similar to natural languages, also contain abundant statistical properties. However, there are also obvious differences that the code of programming language contains rich and clear structural information between programming language and natural language[20]. By extracting the structure information in the code, we can better analyze the source program. Therefore, some works have studied how to capture the code structure information[2, 20, 22].

In [20], Mou et al. parse code into AST and design a novel Tree-Based Convolutional Neural Network (named as TBCNN) to capture code structural information. In TBCNN model, an AST node is first represented as a distributed, real-valued vector so that the (anonymous) features of the symbols are captured. The vector representations are learned by a coding criterion in [22]. Then Mou et al. design a set of subtree feature detectors, called the tree-based convolution kernel, sliding over the entire AST to extract structural information of a program. Thereafter they apply dynamic pooling [26] to obtain information over different parts of the tree. Finally, a hidden layer and an output layer are added.

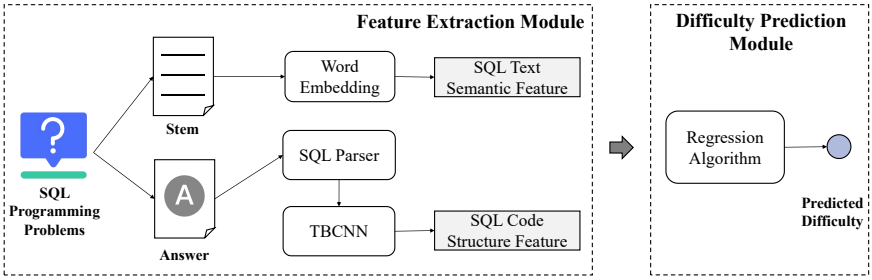


Figure 3: SQL-DP: A difficulty prediction framework for SQL programming problems.

Because the TBCNN model can capture code semantics efficiently, it has become one of the most classical models of code structure feature extraction.

As a programming language, SQL can generate AST through SQL syntax parsers, and then we can use the TBCNN model to extract the structure information of AST, that is, the structure information of SQL code. In short, to obtain more problem information to predict the problem difficulty, except the stem of SQL programming problems, we use the TBCNN model to capture the code structure information from the answers to SQL programming problems.

3.2 Problem Definition

This paper focuses on the difficulty prediction of SQL programming problems. The example of SQL programming problem is shown in Figure 2. The goal of this paper is to train problem difficulty prediction model by using problem stems, answers and real difficulty of SQL programming problems, and then predict the difficulty of new SQL programming problems.

Formally, given the SQL programming problem set $P = \{p_1, p_2, \cdots, p_m\}$ and the corresponding real problem difficulty set $D = \{d_1, d_2, \cdots, d_m\}$, the goal is to use the above data P and D to train a model $\mathcal{M}$, and the trained model $\mathcal{M}$ can estimate the difficulty of new SQL programming problems without test logs. Where P includes the problem stem text set $T = \{t_1, t_2, \cdots, t_m\}$ and the problem answer set $A = \{a_1, a_2, \cdots, a_m\}$. In addition, $p_i = \{t_i, a_i, d_i\}$,$i \in \{1, 2, \cdots, m\}$, p_i represents the SQL programming problem i, t_i represents the problem stem text of p_i, a_i represents the answer to SQL programming problem i, d_i represents the real difficulty of p_i, and m represents the total number of SQL programming problems.

In addition, We obtain the real difficulty of each SQL programming problem from the test logs, followed the previous works[23, 16, 29]. Specifically, we calculate the proportion of incorrect answers by dividing the number of students who have answered the problem incorrectly by the number of students who have responded to the problem[23]. The calculation equation of the difficulty of the problem p_i is as follows:

$$d_i = \frac{s_i}{S_i} \tag{1}$$

where d_i is real difficulty of p_i, s_i represents the number of students who have answered p_i incorrectly, and S_i represents the total number of students who have responded to the problem p_i.

4. SQL-DP FRAMEWORK

This section will describe the difficulty prediction framework of SQL programming problems in detail. As shown in Figure 3, the difficulty prediction framework of SQL programming problems can be divided into two modules: 1) Feature extraction module, which mainly extracts features from the stem and answer of SQL programming problems. The extracted features include stem text semantic features and code structure features; 2) The difficulty prediction module, which uses machine learning to predict the difficulty value of SQL programming problems. Briefly, given the SQL programming problem include stem, answer, and difficulty, we use the feature extraction model to obtain the text semantic features and code structure features from the stems and answers. Then we take the above features and real difficulty as the input of the difficulty prediction module, where the real difficulty is the trained label. Finally, we can use the trained model, the stem and answer of new SQL programming problems to predict the difficulty of new problems, that is, the new SQL programming question without test logs.

4.1 Feature Extraction Module

The feature extraction module consists of SQL text semantic feature extraction module and SQL code structure feature extraction module. The former module uses word embedding techniques to obtain the text semantic features from the stem of SQL programming problems, and the last module extracts SQL code structure features from the answer to problems. The two modules mentioned above will be described in detail below.

4.1.1 SQL Text Semantic Feature Extraction Module

In the task of question difficulty prediction, word embedding technique is often used to obtain the text features of the question, including word2vec, Term Frequency–Inverse Document Frequency (TF-IDF), etc[16, 18, 29]. To extract textual semantic features in SQL programming problems effectively, we adopt various word embedding techniques used in [16] and [29] to extract textual information, including Bag-of-Words (BoW), TF-IDF, and word2vec. Finally, we select the optimal word embedding technique for follow-up experiments.

BoW does not consider the order of words in the sentence.

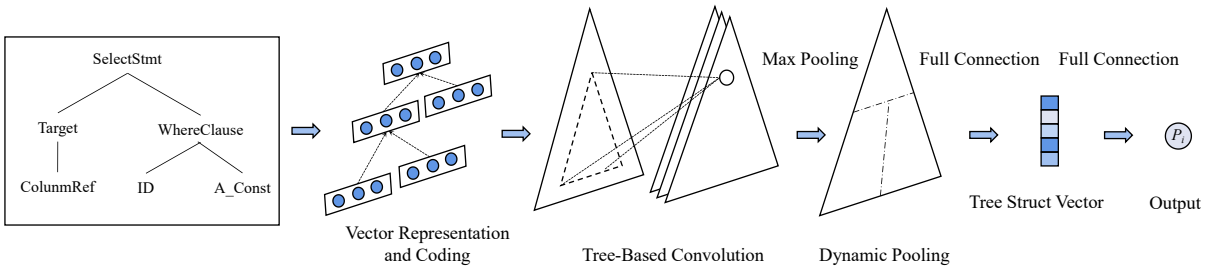


Figure 4: The structure of TBCNN model.

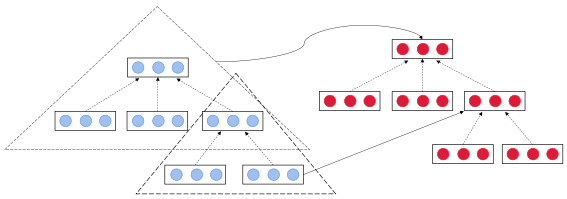


Figure 5: Tree-based convolution.

Still, it only counts the number of occurrences of words, so the value of each position of the text vector calculated by the model is the number of occurrences of the corresponding word. Because the model only records the first occurrence of each word and the number of occurrences of each word, the text features calculated by the model do not contain important information such as the grammar and semantics of the text.

TF-IDF is a word embedding technique used to calculate the importance of each word. The importance of the evaluation word is based on the term frequency and the inverse document frequency of the word appearing in the text. The term frequency counts the number of times each word appears in the text. And the inverse document frequency is used To measure the commonness of each word. The more common words have smaller IDF values.

word2vec is a technique that can efficiently learn the vector representation of text, which can accurately capture the syntactic and semantic word relationships in the text. When using word2vec to extract semantic information in the stem of SQL programming problems, we firstly take all the stem texts of SQL programming problems to train the word2vec model and obtain the word embedding vector of each word or character. Then, we segment the stem, remove the stop word, and replace the remaining words or characters with the corresponding word embedding vector in order to obtain the representation vector of the stem. Next, the input of the difficulty prediction module is a fixed length feature, but the length of words in the different stems is not the same. According to the method of [29], we set the text semantic feature of the stem to a fixed length J. If the length of stem vector is less than J, it is filled with zero. Otherwise, the word behind J is deleted. Finally, we get the text semantic feature vector of SQL programming problem, which can retain certain semantic features.

4.1.2 *SQL Code Structure Feature Extraction Module*

Inspired by [20] and [32], in order to capture the tree structure information of the code as much as possible and enrich the input of the difficulty prediction of SQL programming problems, we apply TBCNN model to capture SQL code structural information. The TBCNN model structure is shown in Figure 4.

TBCNN model includes embedding layer, tree-based convolution layer, dynamic pooling layer, and output layer. TBCNN first converts the abstract syntax tree of SQL code into a vector representation suitable for later calculation. Then, local features are extracted by tree-based convolution, and a new set of feature vectors with the same structure as the input tree are obtained. However, the tree structure of different SQL codes will be different. Therefore, in order to input it into the final full connection layer, the tree structure obtained from the convolution layer is simplified into a fixed vector shape through the dynamic pooling layer. Finally, classification or regression is carried out through the full connection layer. This paper applies TBCNN model to the regression problem, and the training label of the model is the real difficulty of the problem.

Specifically, the input of TBCNN model is the serialized abstract syntax tree (AST). Therefore, the model first uses the pre-training method in [22] to obtain the representation vector of nodes in ASTs. The more specific representation method of node vector can be found in [22].

Then, in the tree-based convolution layer, a set of fixed-depth feature detectors that can slide on the whole AST is designed to extract the structure information of the program. The subtree feature detectors can be viewed as convolution with a set of finite support kernels, so the subtree feature detector is called tree-based convolution. The output of the feature detector is calculated by Equation 2.

$$y = \tanh(\sum_{i=1}^{n} W_{conv,i} \cdot x_i + b_{conv}) \tag{2}$$

where $y \in R^{N_c}$, $x_1, x_2, \cdots, x_n$ is the vector representation

of nodes in the sliding window, and $W_{conv,i} \in R^{N_c \times N_f}$ is the parameter, $b_{conv} \in R^{N_c}$ is the bias, N_c is the number of feature detectors.

In addition, continuous binary tree (as shown in Figure 5) is proposed to handle the different number of child nodes. The convolution layer uses three weight matrices as parameters, including W^t_{conv}, W^l_{conv}, and W^r_{conv}, in which superscript t, l, and r refer to "top", "left", and "right" respectively. The weight matrix of node x_i in the sliding window is a linear combination of W^t_{conv}, W^l_{conv}, and W^r_{conv} (as shown in Equation 3). η^t_i, η^l_i, and η^r_i are the coefficients. In this paper, we use the method in [32] to calculate the coefficient η^t_i, η^l_i, and η^r_i, as shown below:

$$W_{conv,i} = \eta^t_i W^t_{conv} + \eta^l_i W^l_{conv} + \eta^r_i W^r_{conv} \tag{3}$$

$$\eta^t_i = \frac{d_{max} - d_i}{d_{max}} \tag{4}$$

$$\eta^l_i = \begin{cases} \frac{i-1}{n-1}(1-\eta^t_i) & \text{non-leaf} \\ 0.5(1-\eta^t_i) & \text{leaf} \end{cases} \tag{5}$$

$$\eta^r_i = (1-\eta^t_i)(1-\eta^l_i) \tag{6}$$

where d_{max} represents the depth of the sliding window and d_i represents the depth of node i in the sliding window.

After convolution, structural features in an AST are extracted, but the features are still a tree structured set of vectors, which can not be directly fed into the fully connected layer. Therefore, dynamic pooling [26] is applied to reduce the tree into a fixed shape vector.

Finally, a full connection layer and an output layer are added to predict the difficulty of SQL programming problems. The difficulty prediction task of SQL programming problems in this paper belongs to regression problem, inspired by [29], we construct the loss function of the TBCNN model as the Equation 7.

$$L = \frac{1}{n}\sum_{i=1}^{n}(\widetilde{p_i} - d_i)^2 \tag{7}$$

where $\widetilde{p_i}$ and d_i represent the predicted difficulty value and the real difficulty value of the problem p_i respectively.

4.2 Difficulty Prediction Module

The difficulty of SQL programming problems can be predicted using various regression models. Therefore, the difficulty prediction module of SQL programming problems uses a variety of regression models to predict the difficulty of SQL programming problems to obtain the optimal regression model. Regression models used include linear regression (LR) [4], support vector machine (SVM) [7], gradient boosting decision tree (GBDT) [10], random forest (RF) [5] and back propagation (BP) [24] neural networks. The linear regression model is one of the basic regression models commonly used in linear regression problems, but it cannot solve nonlinear distribution problems. The SVM model is common in both linear and nonlinear problems. The RF model is a kind of integrated algorithm with the advantages of difficult overfitting, strong anti-noise ability, and strong interpretability. The GBDT model is an iterative decision tree algorithm that can be used for regression tasks. The BP neural network refers to the multilayer feedforward neural network trained by the BP algorithm. It has strong representation ability and nonlinear mapping ability, but its strong learning representation ability makes it easy to overfit.

Table 1: The statistics of the dataset.

Statistics	Value
# of answer logs	10952
# of students	283
# of SQL programming problems	318
Average answer logs per question	34.4
Average answer logs per student	38.7

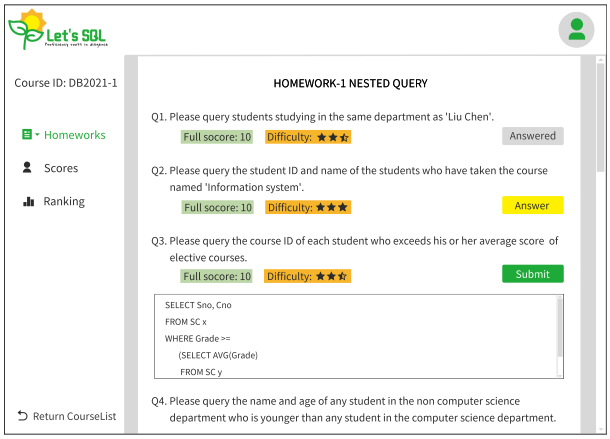


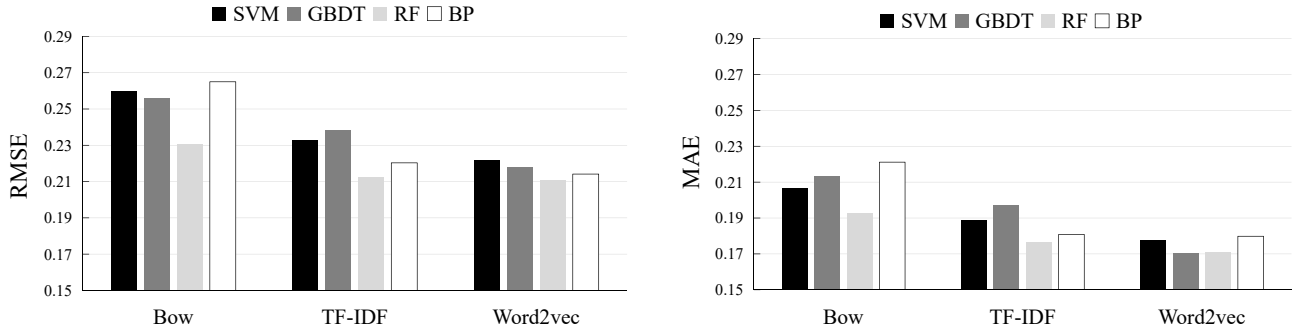
Figure 6: A graph for self-developed SQL online judge system.

The following describes the specific process of the difficulty prediction module:

- Firstly, the text semantic features and code structure features of the extracted SQL programming problems are concatenated together and used as the input of the above regression model. The regression models are trained using the actual difficulty of the SQL programming problems as the training labels.
- Secondly, the model with the best results among the regression models mentioned above is selected as the regression model of the difficulty prediction module of SQL programming problems.
- Thirdly, input the text features and code structure features of the SQL programming problems that need to predict difficulty into the trained regression model. The difficulty of the SQL programming problems can be obtained.

5. EXPERIMENTS

In this section, we first introduce the source of the dataset. Then we raise the evaluation metrics and experimental comparison methods used in the experiment. Next, we present the experimental settings in detail. Finally, we summarize and analyze the experimental results.



(a) RMSE

(b) MAE

Figure 7: Results of text semantic features obtained by different word embedding technologies on different algorithms.

5.1 Dataset Description

The dataset in this paper comes from our self-developed SQL OJ system (as shown in Figure 6). The data collection lasted for two years, involving 306 students from Guangxi University of China. In addition, if only a small count of students have tried to solve a problem, the obtained difficulty of this problem will have severe randomness[23]. Therefore we use the processing method in [23] to process our data. Specifically, we eliminate the problems having no more than ten test logs, and the detail of dataset after processing is shown in Table 1.

5.2 Evaluation Metrics

Followed the previous works[23, 29, 16], we use the evaluation metrics commonly used in question difficulty prediction: *Root Mean Square Error* (RMSE) and *Mean Absolute Error* (MAE). The above two evaluation metrics are widely used in question difficulty prediction to measure the distance between the predicted difficulty value and the actual difficulty value. The smaller the evaluation metric value is, the better performance the results have. The two evaluation metrics are shown in Equation 8 and Equation 9 respectively:

$$\mathrm{RMSE} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\widetilde{p_i} - d_i)^2} \tag{8}$$

$$\mathrm{MAE} = \frac{1}{M}\sum_{i=1}^{M}|\widetilde{p_i} - d_i| \tag{9}$$

where M represents the number of SQL programming problems, and $\widetilde{p_i}$ and d_i represent the predicted difficulty and real difficulty of programming problem i respectively.

5.3 Comparison methods

The difficulty prediction of questions mentioned in the previous related work is not for SQL programming problems. Therefore, we modify C-MIDP model, R-MIDP model, and H-MIDP model of the mathematical questions proposed in [29] to adapt to the SQL programming problems. The above three models first use the word2vec to vectorize the texts of mathematical questions and use the scoring rate of the students in the mathematical questions as the actual difficulty value of the mathematical questions. Then take the real difficulty value of the mathematical questions as the training labels. Finally, input the text representation vector of the mathematical problem into different neural networks for learning. The main processes of the C-MIDP, R-MIDP and H-MIDP models (The structure diagram of C-MIDP, R-MIDP, and H-MIDP is presented in the appendix Table 10) are given below .

- First, use the word2vec model to train all the texts of the mathematical questions, and then the text representation vector of the math questions can be obtained.
- Secondly, the text representation vector of the mathematical question is used as the models' input. The actual difficulty of the question is the scoring rate of each question, and it is used as the training label of the model. During the training process, different models extract different mathematical question information.
- Thirdly, considering that the data in the dataset comes from many different schools. And the student groups in other schools have differences in the knowledge level status, which will cause the scoring rate of the question (i.e., the actual difficulty value) to be affected by the difference in the level of the student group. To eliminate this effect, a context-related training method is proposed. That is, a new loss function is constructed for training the model.
- Finally, for new math questions for which there is no student answer data or insufficient student answer data, the text vector of the new question can be input into the trained model to predict the difficulty of the question.

The C-MIDP, R-MIDP, and H-MIDP models have the same process, and the differences between the three models are: The C-MIDP model uses a multi-layer Convolutional Neural Network to mine different levels of text semantic information from the text of mathematical questions. The R-MIDP model uses the recurrent neural network (RNN) suitable for mining long-range logical relations to mine the sequential logical information of mathematical problems from the text. The H-MIDP model combines the advantages of the

Table 2: TBCNN's hyperparameters.

Hyperparameter	Value
Initial learning rate	0.001
Learning rate decay	0.001
Node embedding dimension	100
Convolutional layers' dimension	50

Table 3: Experimental results of regression model selection

Models	RMSE	MAE
SVM	0.2176	0.1744
GBDT	0.2128	0.1721
RF	0.1977	0.1570
BPNN	0.2022	0.1702

C-MIDP and the R-MIDP and can simultaneously extract crucial semantic information and sequential logic information of mathematical problem text.

5.4 Experimental Setup

Word Embedding: In [16] and [29], Huang et al. and Tong et al. use BoW, TF-IDF, and word2vec technologies to obtain the text semantic features of questions. Following their works, we use the aforementioned methods to process the text semantic features. The Bow and TF-IDF methods are relatively simple, so we only introduce the setting of the word2vec in detail. Specifically, the corpus used for word2vec training is all the stem texts in the dataset. The maximum number of words after word segmentation in the problem stem is 17, so we set the number of words in each problem to 17, and the word vector dimension of each word is 50. Therefore, the stem text vector of each problem has a dimension of 850.

TBCNN Setting: The input of TBCNN model is the AST of programming code. So we first use SQL parser `pglast` [1] to parse the SQL codes, and then we obtain the AST of the codes. After that, we serialize the ASTs and take the serialized ASTs as the input of the TBCNN model. Besides, TBCNN's hyperparameters are shown in Table 2.

Other Setting: All models in the experiment use ten-fold cross-validation to verify the performance of the model. In experiment, the programming language we used is python, and the experiment is configured with 2-core CPU, 8GB memory, 1TB hard disk, and 64-bit Ubuntu operating system.

5.5 Experimental Results

In this section, we run an ablation study to highlight the individual contribution of each module in SQL-DP and compare the SQL-DP proposed in this paper with the C-MIDP, R-MIDP, and H-MIDP model proposed in [29].

5.5.1 Text Semantic Feature Extraction

A variety of word embedding techniques can be used in the SQL-DP framework proposed in this paper, so we need to experiment with different word embedding techniques to obtain the optimal problem text semantic features for subsequent experiments.

Figure 7 shows the results of text semantic features obtained using different word embedding techniques on various machine learning algorithms. But the RMSE of the LR algorithm exceeds 0.5, and the result on the MAE is also poor, so we do not show the results of LR algorithm in Figure 7 in order to a more intuitive display. And the poor experimental results of LR also prove that LR is not competent for difficulty prediction problem. Therefore, we will no longer use the LR algorithm to predict the difficulty of SQL programming problems in the subsequent experiments.

In addition, as shown in Figure 7, the text semantic features extracted by BoW perform the worst in predicting the difficulty of SQL programming problems, while the overall performance of word2vec is the best. This shows that the word2vec is more suitable for extracting text semantic features of problem stems than BoW and TF-IDF for SQL programming problems. Therefore, we will use word2vec to obtain the text semantic features of SQL programming problems in the subsequent experiments.

5.5.2 Regression Model Selection Experiment

SQL-DP can use a variety of regression models to predict the difficulty of SQL programming problems. Therefore, we experiment with multiple regression models to select the best model for subsequent experiments. In the regression model selection experiment, text semantic features and code structure features are used as the input of the regression model at the same time. Table 3 shows the results of using the two features mentioned above as the input of multiple regression models simultaneously. As can be seen from Table 3, when both text semantic features and code structure features are used as the input of the regression model, the results of SQL-DP using the SVM model are the worst, while the effects of the RF model are the best. Therefore, in the subsequent comparative experiment, the SQL-DP framework will use the RF model to predict the difficulty of SQL programming problems.

5.5.3 Comparative Experiment

In this section, we compare SQL-DP with C-MIDP, R-MIDP, and H-MIDP to prove the effectiveness of the difficulty prediction framework of SQL programming problems proposed in this paper. Given the above text semantic feature extraction experiment and ablation experiment results, we choose the SQL-DP framework using word2vec, TBCNN, and RF algorithm to compare with the C-MIDP, R-MIDP, and H-MIDP.

Table 4 shows the experimental results of SQL-DP, C-MIDP, R-MIDP, and H-MIDP. We can see from Table 4 that results of SQL-DP with respect to the two evaluation metrics, i.e., RMSE and MAE, are consistently better than those of C-MIDP, R-MIDP, and H-MIDP models, which verifies the superiority of the proposed SQL-DP in predicting the difficulty of SQL programming problems. Moreover, we are delighted to see from the table that SQL-DP increases the RMSE by 7.23%, compared with the best comparison model H-MIDP,

[1] `https://pglast.readthedocs.io/en/v3/index.html`

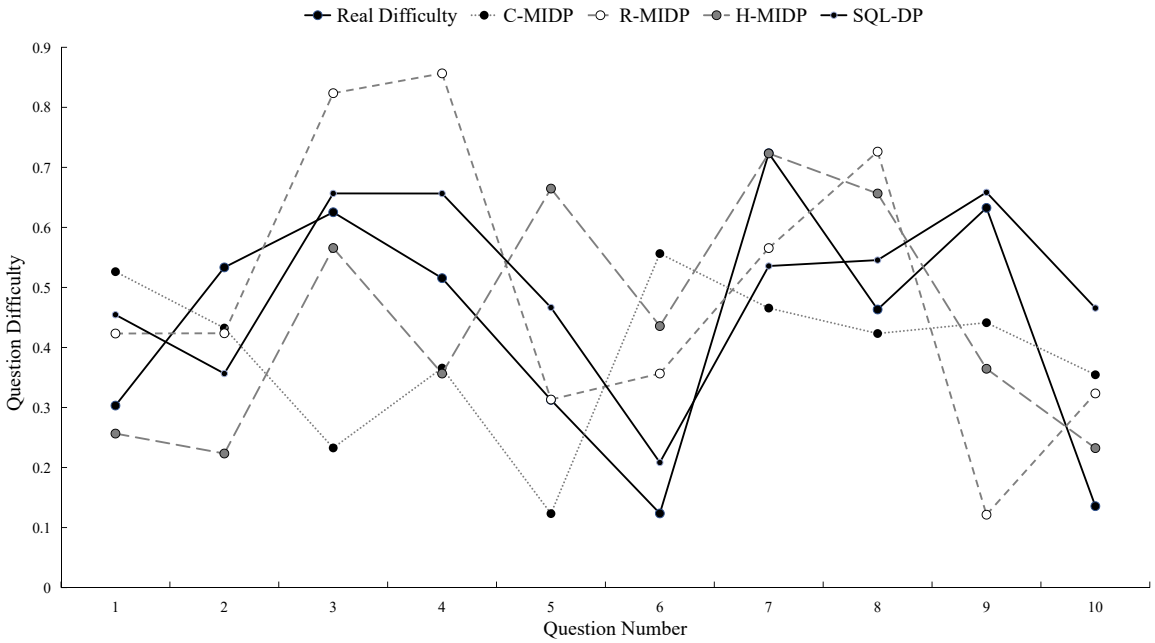


Figure 8: Comparison between the difficulty predicted by four models and the ground truth on ten SQL programming problems.

which is a great progress made under the context of difficulty prediction for SQL programming problems. Because the C-MIDP, R-MIDP, and H-MIDP models only extract information from the text of SQL programming problems and do not extract the code structure information of the answers. Unlike those three comparison models, in SQL-DP, in addition to the text semantic feature extraction module, we can obtain rich semantic information from the stem of SQL programming problems. The code structure extraction module can also obtain rich code structure information from the answers to SQL programming problems. It is the consideration of code structure information in SQL-DP that further enhances the difficulty prediction ability of it towards SQL programming problems, compared with state-of-the-art models.

In addition, we can observe from Table 4 that the H-MIDP model performs best among all state-of-the-art models, while the R-MIDP model performs worst among the three models C-MIDP, R-MIDP, and H-MIDP. The reason may be that the RNN in the R-MIDP model is better at capturing the logical relationship in long sentences. But in the SQL programming problem dataset collected in this paper, the descriptions of the stem of the SQL programming problems are generally short, so the performance of the R-MIDP model on the SQL programming problems is the worst.

To more intuitively see the advantages of our proposed SQL-DP in predicting the difficulty of SQL programming problems, we randomly select 10 problems in the test set. And we test them with the trained C-MIDP model, R-MIDP model, H-MIDP model, and our proposed SQL-DP. After that, we use a broken line diagram (as shown in Figure 8) to show the distance between the predicted problem difficulty of the above models and the real problem difficulty. As shown in Figure 8, we can see that the prediction difficulty of SQL-DP for the selected SQL programming problems is closer to the real problem difficulty than the C-MIDP, R-MIDP, and H-MIDP models.

Table 4: Comparative experimental results.

Models	RMSE	MAE
C-MIDP	0.2167	0.1603
R-MIDP	0.2228	0.1785
H-MIDP	0.2131	0.1578
SQL-DP	0.1977	0.1570

Table 5: Ablation experimental results of SQL-DP.

Feature	RMSE	MAE
Text semantic feature	0.2106	0.1711
Code structure feature	0.2124	0.1721
Text semantic + Code structure	0.1977	0.1570

5.5.4 Ablation Experimental

To get deep insights into the contributions of various modules in the SQL-DP framework proposed in this paper, we also conduct some ablation prediction outcomes.

As shown in Table 5, we can observe a performance decrease by removing the SQL text semantic feature extraction module or the SQL code structure feature extraction module. Besides, we can also see that the overall performance of the SQL text semantic feature extraction module is similar to that of the SQL code structure feature extraction module. This observation shows that both SQL code structure fea-

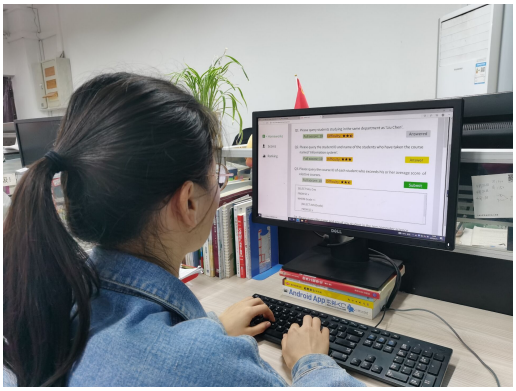


Figure 9: Scenario of a student using self-developed SQL OJ system.

tures and SQL text semantic features play an essential role in the difficulty prediction of SQL programming problems.

5.5.5 *Application*

Given the effectiveness of the SQL-DP framework proposed in this paper, we use the trained SQL-DP to predict the difficulty of new SQL programming problems. That is, the problem without student test logs. And apply it to our self-developed SQL OJ system, as shown in Figure 6. Specifically, we map the problem difficulty value from 0 to 1 to 5 stars. Among them, the problem difficulty value from 0 to 0.1 is half a star, value from 0.1 to 0.2 is one star, and so on. Figure 9 shows a student practicing SQL programming problems using our self-developed SQL OJ system. The student can choose to do simple SQL programming problems first according to the problem difficulty labels or challenge herself to choose more difficult SQL programming problems in our system.

6. CONCLUSIONS

Conclusions. In this paper, we propose SQL-DP, a novel framework that automatically predicts difficulty of SQL programming problems. SQL-DP makes full use of the information in problem stems and problem answers in the form of SQL codes, based on which the difficulty values of SQL programming problems can be effectively estimated using machine learning techniques. Besides, we organized many tests of SQL programming problems in real teaching practice, which last for two years and involve seven different undergraduate classes in Guangxi University of China, and collected a SQL dataset containing materials and student answering logs of hundreds of SQL programming problems. Experimental results over our collected physical-world SQL dataset show that the proposed SQL-DP gains apparently better prediction performance towards SQL programming problems, compared with state-of-the-art solutions.

Generalization. Though SQL-DP is discussed for the difficulty prediction of SQL programming problems, it can be easily generalized to address the difficulty prediction of programming problems using other languages (such as C and Java), since the consideration of structure information of problem answers is also very important for these programming problems.

Future Works. Two important directions for future works can be considered. First, we will consider more features of SQL programming problems, such as equivalent answers, SQL concepts (e.g., nested queries, multiple tables), etc. Second, to support the difficulty prediction task of programming problems corresponding to more types of programming languages, we will modify and adapt the proposed SQL-DP framework, including the design or use of neural network layers.

Related Resources. To better promote related study of SQL programming problems, the source code of the proposed SQL-DP framework and partial of our collected SQL dataset used in the experiment are all released and can be assessed though the link below: `https://github.com/SQL-DP/SQL-DP`

7. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. 62067001), the Projects of Higher Education Undergraduate Teaching Reform Project in Guangxi (Nos. 2017JGZ103 and 2020JGA116), Innovation Project of Guangxi Graduate Education (No. JGY2021003), and the Special funds for Guangxi BaGui Scholars. This work is partially supported by the Guangxi Natural Science Foundation (No. 2019JJA170045). We would like to thank Dr. Wu, Dr. Tian, and Mr. Zhang for providing data from their course teaching practices. Finally, thank Aetf for providing some source code.

8. REFERENCES

[1] S. AlKhuzaey, F. Grasso, T. R. Payne, and V. A. M. Tamma. A systematic review of data-driven approaches to item difficulty prediction. In *Proceedings of the 22nd International Conference on Artificial Intelligence in Education*, pages 29–41. Springer, June 2021.

[2] N. D. Q. Bui, Y. Yu, and L. Jiang. Treecaps: Tree-based capsule networks for source code processing. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pages 30–38. AAAI Press, February 2021.

[3] Y. V. Chon and T. Shin. Item difficulty predictors of a multiple-choice reading test. *ENGLISH TEACHING*, 65(4):257–282, December 2010.

[4] D. R. Cox. Corrigenda: The regression analysis of binary sequences. *Journal of the Royal Statistical Society*, 21(1):238, 1959.

[5] A. Cutler, D. R. Cutler, and J. R. Stevens. *Random forests*. Springer US, Boston, 2004.

[6] DeVellis and F. Robert. Classical test theory. *Medical Care*, 44(3):S50–9, December 2006.

[7] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, pages 155–161. MIT Press, December 1996.

[8] Y. H. El Masri, S. Ferrara, P. W. Foltz, and J. A. Baird. Predicting item difficulty of science national curriculum tests: The case of key stage 2 assessments. *The Curriculum Journal*, 28(1):59–82, November 2017.

[9] X. FAN. Item response theory and classical test theory: An empirical comparison of their item/person

statistics. *Educational and Psychological Measurement*, 58(3):357–381, June 1998.

[10] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, November 2001.

[11] H. Fu-Yuan, L. Hahn-Ming, C. Tao-Hsing, and S. Yao-Ting. Automated estimation of item difficulty for multiple-choice tests: An application of word embedding techniques. *Information Processing & Management*, 54(6):969–984, January 2018.

[12] G. Y. Han and X. Z. Li. An intelligent test paper generation algorithm based on adjustment of overall difficulty degrees. *Applied Mechanics and Materials*, 411-414(APR.):2879–2882, September 2013.

[13] A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. T. Devanbu. On the naturalness of software. In *34th International Conference on Software Engineering*, pages 837–847. IEEE Computer Society, June 2012.

[14] P. W. Holland and D. T. Thayer. An alternate definition of the ets delta scale of item difficulty. *ETS Research Report Series*, 1985(2):i–10, December 1985.

[15] P. Hontangas, V. Ponsoda, J. Olea, and S. L. Wise. The choice of item difficulty in self-adapted testing. *European Journal of Psychological Assessment*, 16(1):3–12, 2000.

[16] Z. Huang, Q. Liu, E. Chen, H. Zhao, M. Gao, S. Wei, Y. Su, and G. Hu. Question difficulty prediction for READING problems in standard tests. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence*, pages 1352–1359. AAAI Press, February 2017.

[17] P. Klein, S. Tirthapura, D. Sharvit, and B. Kimia. A tree-edit-distance algorithm for comparing simple, closed shapes. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, page 696–704. Society for Industrial and Applied Mathematics, January 2000.

[18] L. Lin, T. Chang, and F. Hsu. Automated prediction of item difficulty in reading comprehension using long short-term memory. In *International Conference on Asian Language Processing*, pages 132–135. IEEE, November 2019.

[19] A. Lino, A. Rocha, L. Macedo, and A. Sizo. Application of clustering-based decision tree approach in sql query error database. *Future generation computer systems*, 93(APR.):392–406, April 2019.

[20] L. Mou, G. Li, L. Zhang, T. Wang, and Z. Jin. Convolutional neural networks over tree structures for programming language processing. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 1287–1293. AAAI Press, February 2016.

[21] I. Pandarova, T. Schmidt, J. Hartig, A. Boubekki, R. D. Jones, and U. Brefeld. Predicting the difficulty of exercise items for dynamic difficulty adaptation in adaptive language tutoring. *Int. J. Artif. Intell. Educ.*, 29(3):342–367, 2019.

[22] H. Peng, L. Mou, G. Li, Y. Liu, L. Zhang, and Z. Jin. Building program vector representations for deep learning. In *Proceedings of the 8th International Conference on Knowledge Science, Engineering and Management*, pages 547–553. Springer, October 2015.

[23] Z. Qiu, X. Wu, and W. Fan. Question difficulty prediction for multiple choice problems in medical exams. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 139–148. ACM, November 2019.

[24] D. E. Rumelhart and J. L. Mcclelland. *Parallel distributed processing*. MIT Press, Cambridge, 1986.

[25] M. Sano. Improvements in automated capturing of psycho-linguistic features in reading assessment text. In *The annual meeting of National Council on Measurement in Education*, pages 1–27. National Council on Measurement in Education, April 2016.

[26] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pages 801–809. Curran Associates Inc., December 2011.

[27] Y. Susanti, H. Nishikawa, T. Tokunaga, and O. Hiroyuki. Item difficulty analysis of english vocabulary questions. In *Proceedings of the 8th International Conference on Computer Supported Education*, page 267–274. SciTePress, April 2016.

[28] T. Taipalus. *Teaching tip*: A notation for planning SQL queries. *J. Inf. Syst. Educ.*, 30(3):160–166, Winter 2019.

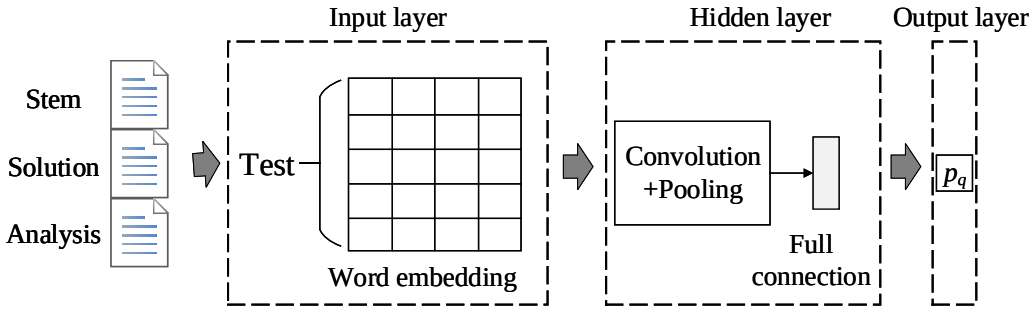
[29] W. Tong, F. Wang, Q. Liu, and E. Chen. Data driven prediction for the difficulty of mathematical items. *Journal of Computer Research and Development*, 56(5):1007–1019, May 2019.

[30] J. D. L. TORRE. Dina model and parameter estimation: A didactic. *Journal of Educational and Behavioral Statistics*, 34(1):115–130, March 2009.

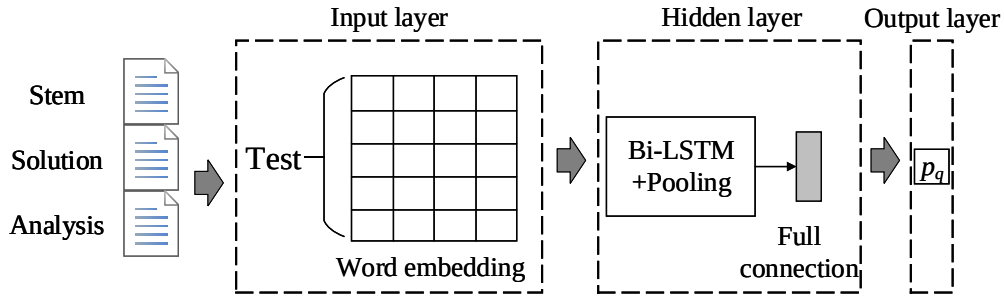
[31] Z. Wu, M. Li, Y. Tang, and Q. Liang. Exercise recommendation based on knowledge concept prediction. *Knowledge-Based Systems*, 210:106481, December 2020.

[32] P. Yu and S. Wang. Deep tree: Sql injection detection by the power of deep learning. `https://github.com/Aetf/tensorflow-tbcnn/blob/master/misc/deeptree.pdf`, 2017.

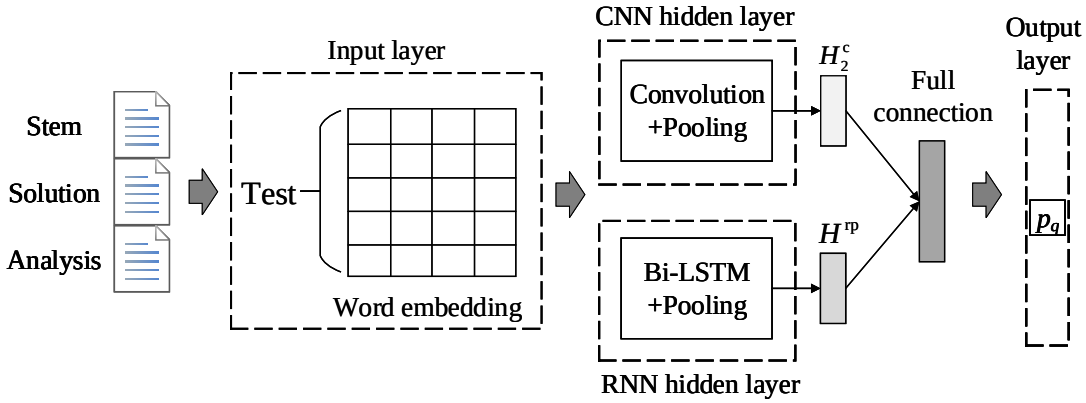
APPENDIX



(a) Structure diagram of C-MIDP model



(b) Structure diagram of R-MIDP model



(c) Structure diagram of H-MIDP model

Figure 10: Structure diagram of C-MIDP, R-MIDP, and H-MIDP model.