

**Automated Scoring for Short Questions with Deep Learning**

Submitted by

Michelle Vanessa

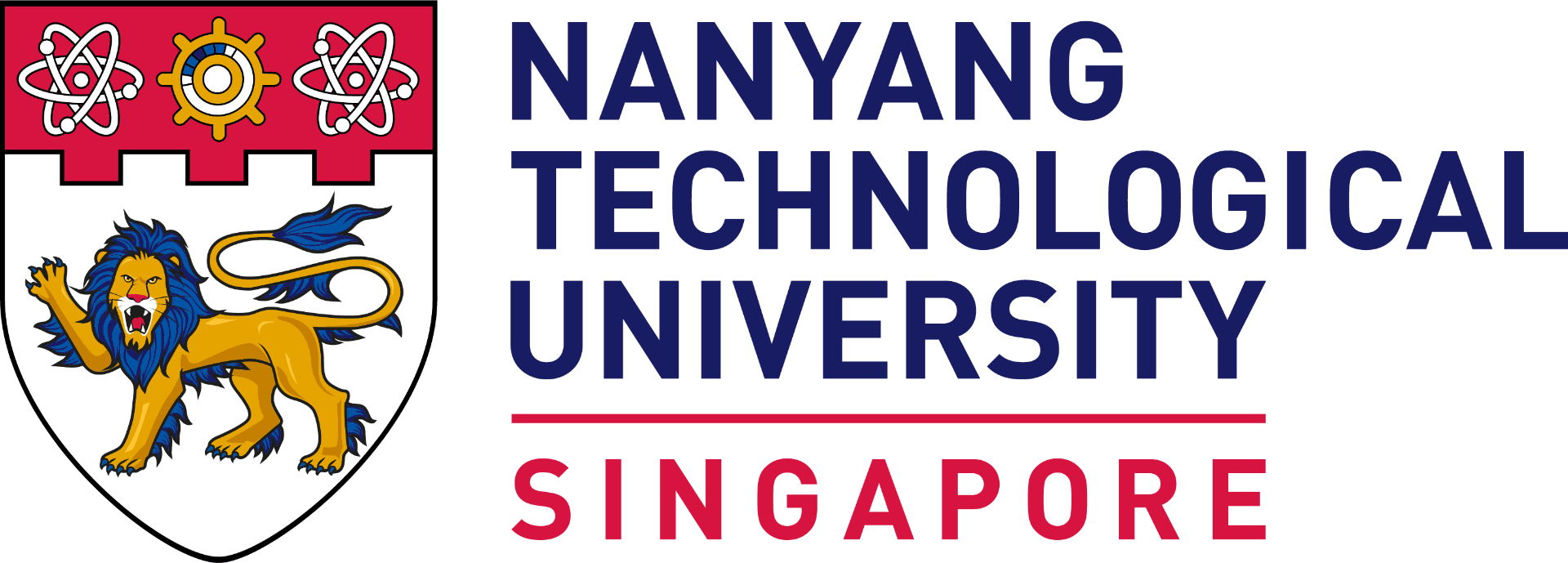
U1620071L

Project Supervisor

Assoc Prof. Hui Siu Cheung

School of Computer Science and Engineering

AY2019/2020



**SCE19-0305**

**Automated Scoring for Short Questions with Deep Learning**

Submitted in Partial Fulfilment of the Requirement for the Degree of Bachelor of Engineering (Computer Science)

Submitted by

Michelle Vanessa

U1620071L

Project Supervisor

Assoc Prof. Hui Siu Cheung

School of Computer Science and Engineering

AY2019/2020

# Abstract

This report proposes an improved approach of short-answered questions scoring with deep learning, namely Siamese Bidirectional LSTM model with feature engineering. The project will conduct several experiments in order to obtain optimal performance of the model. The scope of the questions is limited to Data Structures questions, and a dataset known as Mohler dataset is used to train the model in this experiment. Eventually, the model will be deployed into a web server for data collection so that the model can be further trained and for self-practice purposes for computer science students.

# Acknowledgements

The author would like to express her gratitude to various individuals for their continuous assistance. This project would not have been possible without their support.

Firstly, the author would like to extend her deepest gratitude to Assoc Prof Hui Siu Cheung, the author’s final year project supervisor, for his guidance, encouragement, and constructive feedbacks throughout the year and to her parents who have supported her throughout her journey in Nanyang Technological University Singapore. Without them, this project would not have been as successful.

Lastly, the author wishes to thank all her friends who have supported during her time in Nanyang Technological University Singapore and have been there for her when she needed encouragement. Without their faith in her, the author would not have been able to complete this project smoothly.

Table of Contents

[Abstract 2](#_Toc35473238)

[Acknowledgements 3](#_Toc35473239)

[Table of Contents 4](#_Toc35473240)

[List of Figures 7](#_Toc35473241)

[List of Equations 9](#_Toc35473242)

[List of Tables 10](#_Toc35473243)

[Chapter 1: Introduction 11](#_Toc35473244)

[1.1 Background 11](#_Toc35473245)

[1.2 Motivation 11](#_Toc35473246)

[1.3 Objectives 11](#_Toc35473247)

[Chapter 2: Literature Review 13](#_Toc35473248)

[2.1 Past Works 13](#_Toc35473249)

[2.2 Multilayer Perceptron 13](#_Toc35473250)

[2.3 Recurrent Neural Network (RNN) 14](#_Toc35473251)

[2.4 Batch Normalization 14](#_Toc35473252)

[2.5 Dropout 14](#_Toc35473253)

[2.6 Question Demoting 14](#_Toc35473254)

[Chapter 3: Existing Approach 16](#_Toc35473255)

[3.1 Architecture 16](#_Toc35473256)

[3.1.1 Embedding Layer 16](#_Toc35473257)

[3.1.2 Vertical Layers 17](#_Toc35473258)

[3.1.3 Concatenation Layer 19](#_Toc35473259)

[3.1.4 Perceptron Layers 19](#_Toc35473260)

[3.1.5 Output Layer 19](#_Toc35473261)

[3.2 Deep Learning Models 19](#_Toc35473262)

[3.2.1 Multilayer perceptron 19](#_Toc35473263)

[3.2.2 Bidirectional LSTM Unit 19](#_Toc35473264)

[3.3 Techniques 22](#_Toc35473265)

[3.3.1 Batch Normalization 22](#_Toc35473266)

[3.3.2 Dropout 23](#_Toc35473267)

[Chapter 4: Experimental Evaluation 24](#_Toc35473268)

[4.1 Dataset 24](#_Toc35473269)

[4.2 Evaluation Metrics 24](#_Toc35473270)

[4.2.1 Pearson Correlation Coefficient 24](#_Toc35473271)

[4.2.2 Rooted Mean Square Error 25](#_Toc35473272)

[4.2.3 Mean Absolute Error 25](#_Toc35473273)

[4.3 Data Augmentation 26](#_Toc35473274)

[4.4 Evaluation 26](#_Toc35473275)

[4.4.1 Implementation 26](#_Toc35473276)

[4.4.2 Model Selection 26](#_Toc35473277)

[4.4.3 Training 27](#_Toc35473278)

[4.5 Performance of Existing Model 27](#_Toc35473279)

[4.6 Improving Model Performance 28](#_Toc35473280)

[4.6.1 Recurrent Batch Normalization 29](#_Toc35473281)

[4.6.2 Regularization 29](#_Toc35473282)

[4.6.3 Question Demoting 30](#_Toc35473283)

[4.6.4 Classification Model 30](#_Toc35473284)

[4.7 Experiment Results 32](#_Toc35473285)

[4.7.1 Recurrent Batch Normalization Results 32](#_Toc35473286)

[4.7.2 Regularization Results 32](#_Toc35473287)

[4.7.3 Question Demoting 34](#_Toc35473288)

[4.7.4 Classification Results 36](#_Toc35473289)

[Chapter 5: System Implementation 38](#_Toc35473290)

[5.1 Software Architectural Patterns 38](#_Toc35473291)

[5.2 System Interface 38](#_Toc35473292)

[5.2.1 General 38](#_Toc35473293)

[5.2.2 Admin 39](#_Toc35473294)

[5.2.3 Student 44](#_Toc35473295)

[Chapter 6: Conclusion 48](#_Toc35473296)

[Reference 49](#_Toc35473297)

[Appendix A: Experiment Results (Regression Model) 52](#_Toc35473298)

List of Figures

[Figure 1 Network architecture illustration 15](#_Toc35473299)

[Figure 3 Multilayer perceptron (tokenization) 16](#_Toc35473300)

[Figure 4 Multilayer perceptron (feature engineering) 17](#_Toc35473301)

[Figure 5 LSTM Unit 20](#_Toc35473302)

[Figure 6 LSTM network 20](#_Toc35473303)

[Figure 7 Bidirectional LSTM network 21](#_Toc35473304)

[Figure 8 Dropout on neural network with 2 hidden layers 22](#_Toc35473305)

[Figure 9 Pearson correlation coefficient 24](#_Toc35473306)

[Figure 10 Training loss of original model 27](#_Toc35473307)

[Figure 11 Validation loss off original model 27](#_Toc35473308)

[Figure 12 Training loss of optimal model 34](#_Toc35473309)

[Figure 13 Validation loss of optimal model 34](#_Toc35473310)

[Figure 14 Sign up page 38](#_Toc35473311)

[Figure 15 Log in page 38](#_Toc35473312)

[Figure 16 Admin questions list 39](#_Toc35473313)

[Figure 17 Admin question details 39](#_Toc35473314)

[Figure 18 Add questions 40](#_Toc35473315)

[Figure 19 Admin posts list 40](#_Toc35473316)

[Figure 20 Admin post details 41](#_Toc35473317)

[Figure 21 Admin answers list 41](#_Toc35473318)

[Figure 22 Add answers 42](#_Toc35473319)

[Figure 23 Model details 42](#_Toc35473320)

[Figure 24 Students list 43](#_Toc35473321)

[Figure 25 Student details 43](#_Toc35473322)

[Figure 26 Student questions list 44](#_Toc35473323)

[Figure 27 Student answer question page 44](#_Toc35473324)

[Figure 28 Student posts list 45](#_Toc35473325)

[Figure 29 Student account details 45](#_Toc35473326)

[Figure 30 Student edit account 46](#_Toc35473327)

List of Equations

[Equation 1 Distance between two vectors 19](#_Toc35473328)

[Equation 2 Forget gate 20](#_Toc35473329)

[Equation 3 Sigmoid function 20](#_Toc35473330)

[Equation 4 Input gate sigmoid layer 20](#_Toc35473331)

[Equation 5 Input gate tanh layer 20](#_Toc35473332)

[Equation 6 Tanh function 20](#_Toc35473333)

[Equation 7 Output gate 20](#_Toc35473334)

[Equation 8 Cell state 21](#_Toc35473335)

[Equation 9 LSTM output 21](#_Toc35473336)

[Equation 10 Mini-batch mean 22](#_Toc35473337)

[Equation 11 Mini-batch variance 22](#_Toc35473338)

[Equation 12 Normalization 22](#_Toc35473339)

[Equation 13 Scaling and shifting 22](#_Toc35473340)

[Equation 14 Pearson correlation coefficient 24](#_Toc35473341)

[Equation 15 Covariance 24](#_Toc35473342)

[Equation 16 Standard deviation 24](#_Toc35473343)

[Equation 17 RMSE 25](#_Toc35473344)

[Equation 18 MAE 26](#_Toc35473345)

[Equation 19 L2 regularization term 29](#_Toc35473346)

[Equation 20 Cost function with L2 regularization 29](#_Toc35473347)

[Equation 21 Target output discretization 31](#_Toc35473348)

List of Tables

[Table 1 Performance metrics of existing approach 24](#_Toc35380235)

[Table 2 Experiment results on recurrent batch normalization 29](#_Toc35380236)

[Table 3 Experiment results of weight decay on LSTM units 29](#_Toc35380237)

[Table 4 Experiment results of weight decay on perceptron 30](#_Toc35380238)

[Table 5 Experiment results of dropout on LSTM layers 30](#_Toc35380239)

[Table 6 Experiment results of dropout on perceptron layers 31](#_Toc35380240)

[Table 7 Experiment results on regularization methods 31](#_Toc35380241)

[Table 8 Experiment results on regularizations 31](#_Toc35380242)

[Table 9 Experiment results on question demoting 32](#_Toc35380243)

[Table 10 Experiment results on optimal model with question demoting 32](#_Toc35380244)

[Table 11 Experiment results on classification model 33](#_Toc35380245)

[Table 12 Experiment results on classification model with question demoting 33](#_Toc35380246)

[Table 13 Experiment results on classification model with regularization and question demoting 33](#_Toc35380247)

# Introduction

## Background

One of the main tasks of an educator is to assess the understanding of their students. It can be measured through various assessments, which requires the educator to manually evaluate and grade students’ responses. Time taken to do such activity depends on the open-endedness of the assessment since answers to open-ended questions are more varied, and thus, additional time is required to assess the answers. Hence, multiple choice questions will take shorter time to score compared to essay questions. Moreover, scoring open-ended questions is susceptible to the grader’s subjectivity as there is no right or wrong answer.

Automated grading system would help speed up scoring process and would be able to objectively grade student responses, thus, eliminating the disadvantages of manual human scoring.

## Motivation

The education sector would benefit greatly from the automated essay scoring. Despite the fact that multiple choice questions scoring is now mostly done by machine, automated essay scoring is still not widely used even though it has been a research topic for some time now.

Most educational system still requires human to manually grade them, which consumes a lot of the educators’ time. With the help of the automated essay scoring, the grading process would be shortened [1], and educators would be able to reduce the time they spend evaluating assessments, hence, increasing educators’ productivity on other activities. Moreover, the automated essay scoring has numerous advantages over manual scoring, such as its objectivity and consistency.

* 1. Objectives

The objective of this project is to improve the existing short answer scoring approach, namely the Siamese Bidirectional LSTM model. The approach will be improved by applying various methods on the model and on the dataset. However, the scope of the questions is limited to Data Structures questions only so that the answers could be more topic specific. With the limited scope of the questions, the answers will have similar features, and hopefully, the model can pick up important features from the answers more easily.

The model would later on be deployed into a system, and the system would be implemented in an educational setting, where it could help facilitate the scoring process of short-answered data structures related C programming questions, which, in this case, is to assist scoring of CX1007 Data Structures course as a part of students’ self-practice and performance evaluation.

# Literature Review

* 1. Past Works

The essay grading issue has been addressed since years ago. One of the earliest approach to the problem is a method known as C-rater [2], which was developed by Educational Testing Service, also widely known as ETS, to measure students’ understanding based on their responses to short-answer questions. The method compares the syntactical characteristics of a sentence to a collection of correct ones. However, it is ignoring the difference between passive and active voice, such as “you need two plants” and “ two plants are needed”.

More recent work has analysed the difference between corpus-based and knowledge-based measures of text similarity, and it was shown using Pearson’s correlation coefficient that corpus-based measure (Lexical Semantic Analysis) performs the best among other approaches. It also introduced new technique which is similar to pseudo-relevance feedback to address the problem where there is more than one correct answer [3].

Although some works use Pearson’s correlation coefficient to compare student and reference answers, another experiment [4] showed that similarity measure using Cosine coefficient produce the best result. Cosine similarity measure has the highest accuracy rate compared to other measures, namely Jaccard coefficient and Dice coefficient.

One of the works that use Cosine similarity is the paragraph embeddings [5], which focused on short answer scoring. Answers are considered short “if its length approximately ranges from one phrase to one paragraph”. The word embedding vectors from the answer are combined using average, sum, or other methods, then, using the calculated vectors, new vectors are generated using paragraph embedding model. Cosine coefficient will be used to compare the paragraph vectors.

* 1. Recurrent Neural Network (RNN)

RNNs are designed to retain information from previous time frames so that the patterns found in the past information can be used to predict the future patterns. Such method is known as long-term dependencies, and RNNs are designed to handle that. However, in practice, RNNs are unable to learn the dependencies, so a new approach, Long Short-Term Memory, was introduced to address this issue [6].

Long Short-Term Memory unit is a type of Recurrent Neural Network. As mentioned in the previous section, the information in this unit goes back in time, as opposed to the traditional multilayer perceptron. It was first introduced in 1997 due to the inability of the conventional approach at that time to prevent the information going backwards to blow up or vanish [7]. To avoid the aforementioned problem, a forget gate was added to the unit so that insignificant information can be discarded.

* 1. Multilayer Perceptron

Multilayer perceptron, a class of feedforward neural network, is a deep learning model that approximates some function by learning parameter so that it could generate the best result. Information flows through the layers of neurons, the first one, being called first layer, until the last one, the output layer. Between the first and output layer, there are more layers, called hidden layers. In contrast to Recurrent Neural Network, information in multilayer perceptron never goes backwards and only goes forward in time [8].

* 1. Batch Normalization

With the discovery of more complex deep learning approaches, the demand of more advanced neural network training techniques became higher. Motivated by the uneven distribution of neural network layers’ input during training, batch normalization was introduced in 2015 [9]. The technique is designed to balance the distribution of the inputs by reducing internal covariate shift (ICS), which is the change in the distribution of the input variables in training and test data [10].

* 1. Dropout

Despite the breakthrough of machine learning techniques, overfitting is still a prominent issue in deep learning, especially when the size of training set is very small compared to the complexity of the network. In this case, dropout can be used to avoid the problem. The concept of dropout is to randomly drop neurons from the network, so that dropped neurons will be omitted during training, and hence reducing the complexity of the network [11]. Therefore, the network would be able to learn information that is significant to produce the correct output [12].

* 1. Question Demoting

Another work proposed a new technique named question demoting [13]. The technique removes any words that occur in the question from both the student and gold standard answers. This technique is implemented to eliminate the possibility that students who repeat words from the question in their answers get high score as it does not reflect their understanding in the topic.

# Existing Approach

The existing approach automates grading of short answer using Siamese bidirectional LSTM-based regression. This chapter discusses the deep learning methods implemented in the approach.

* 1. Architecture

This approach combines several neural network architectures, and together they create the overall model architecture. The model consists of several layers, which are shown in Figure 1 Network architecture illustration.

A screenshot of a cell phone

Description automatically generated

Figure 1 Network architecture illustration

* + 1. Embedding Layer

The first layer of the network is embedding layer. This layer takes in a sequence of words as input, and each word will be mapped into high-dimensional vectors representing the respective words, also known as word embedding. In word embedding, similar words have similar values, and vector operations can be performed to obtain other words’ vector values.

In this layer, Word2Vec is used to compute the vector representations of each word . It is one of word embedding model architectures that utilises Skip-gram model and negative sampling [14]. The words are represented in such a way that the result of the vector operations reflects the linguistic patterns of the words. For example, the operation “Madrid” – “Spain” + “France” will produce a vector close to the vector representation of the word “Paris” [15].

The input of this layer will be the answers, while the output will be the mapped 300-dimensional word vectors of the input. This layer will be used twice, for student answers and reference answers.

* + 1. Vertical Layers

This layer consists of several vertical layers that is independent from each other, and each has its specific functions. Following are the details of the layers as shown in Figure 2 from left to right.

* + - 1. Multilayer perceptron (tokenization)

This layer consists of 2 layers of neurons, both of 50 neurons with sigmoid activation function. It receives 3 integers. The first integer is number of words of the reference answer, the second is number of the student answer, and the last one is number of words that exist both in the student and reference answers. To obtain these numbers, the answers are first tokenized using NLTK word tokenizer. Tokenization is a process where a sequence of words is separated into smaller parts called tokens. [16].

A close up of a logo

Description automatically generated

Figure 2 Multilayer perceptron (tokenization)

* + - 1. Multilayer perceptron (feature engineering)

This layer takes input of 5 set of data obtained from feature engineering. Feature engineering is a process where a certain data is produced by transforming raw data so that it could be used by the model to better learn the pattern of the given data [17]. In this case, length of the student answer (number of characters), ratio of the length of the reference answer and length of the student answer, number of words in the student answer, and the number of unique words in the student answer are used.

The input is then processed through 4 layers of 125 neurons with sigmoid activation function.

A picture containing control

Description automatically generated

Figure 3 Multilayer perceptron (feature engineering)

* + - 1. LSTM unit (reference answer)

This layer receives input from embedding layer, which is the embedded student answers. It will then be propagated through a bidirectional LSTM unit and through a layer of 50 neurons with sigmoid activation function.

* + - 1. LSTM unit (comparison with student answer)

This layer has the same architecture as the previous layer, but it has different inputs. Instead of receiving input straight from the embedding layer, this layer receives a vector of comparison between student answer and reference answer. The two sentences are compared using the distance of the two vectors. The distance is calculated using the equation below, where is the reference answer of question , and is the student answer j of question . *v(X)* is the word vector of sentence *X*.

Equation 1 Distance between two vectors

Word2Vec groups similar words together [15], so words with similar meaning has similar vector values. Hence, the distance of the two vectors should be small if the sentences have similar meaning.

* + 1. Concatenation Layer

In this layer, the output from vertical layers are merged to be processed further in the following layers.

* + 1. Perceptron Layers

The concatenated vector is then propagated through a multilayer perceptron. The multilayer perceptron has 4 layers, where the first 3 layers consist of 125 neurons, and the last one consists of 25 neurons.

A close up of a black background

Description automatically generated

Figure 4 Perceptron layers

* + 1. Output Layer

The final layer of the network consists of 1 neuron with linear activation function. This neuron outputs the final predicted score.

* 1. Deep Learning Models

There are two main neural network architectures that are used in this approach, namely multilayer perceptron and bidirectional LSTM unit.

* + 1. Multilayer perceptron

In this approach, multilayer perceptron is combined with other model to form the overall architecture. The output of the perceptron is concatenated with the output from other models before going through another chain of multilayer perceptron.

* + 1. Bidirectional LSTM Unit

The basic LSTM unit consists of a cell state that modulates information through the unit, and three gates (forget, input, and output) [6]. Following is the functions used in the unit, where is the input and the output.

* + - 1. Forget gate

Forget gate will determine whether the information should be removed or not. When it is 0, nothing will go through.

Equation 2 Forget gate

Where is sigmoid function.

Equation 3 Sigmoid function

* + - 1. Input gate
         1. Sigmoid layer

This layer will determine which values to update.

Equation 4 Input gate sigmoid layer

* + - * 1. Tanh layer

This layer will produce vector of new candidate values.

Equation 5 Input gate tanh layer

Where is tanh function.

Equation 6 Tanh function

* + - 1. Output gate

This gate will allow memory cell to have effect on other neurons.

Equation 7 Output gate

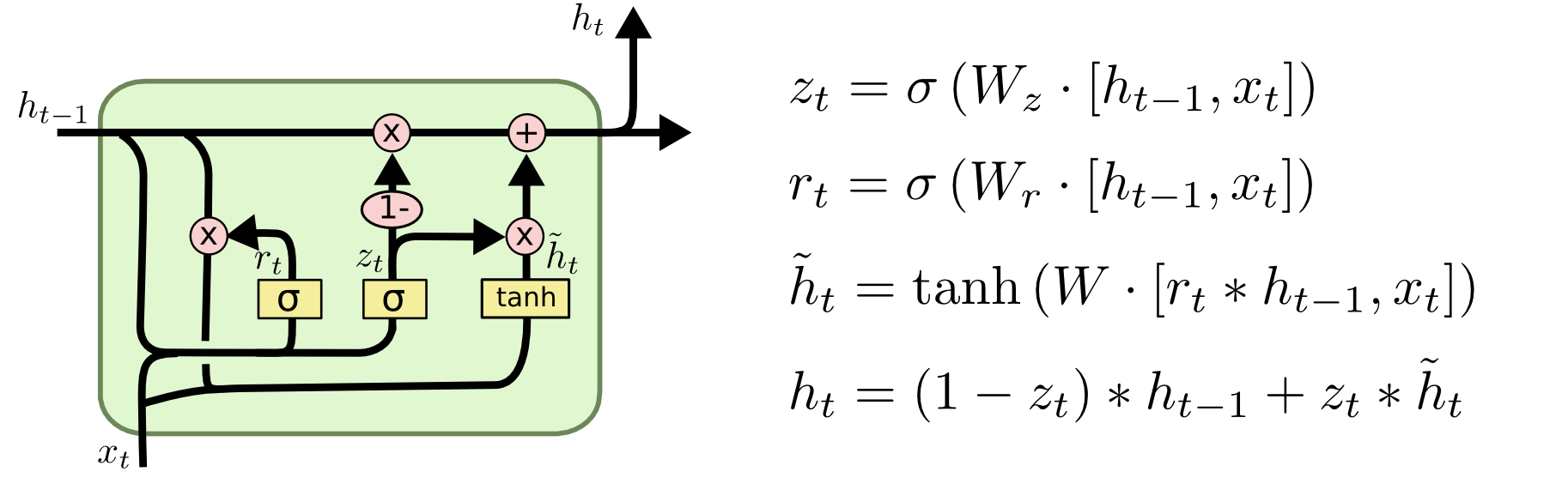


Figure 5 LSTM Unit

Afterwards, the cell state is updated using equation below, where is element-wise product.

Equation 8 Cell state

The output value is calculated using the equation below.

Equation 9 LSTM output

However, this approach uses bidirectional LSTM, which means the neurons are split into two directions, forward and backward. This method will enable the effective usage of both past and present information for a specific time frame [18].

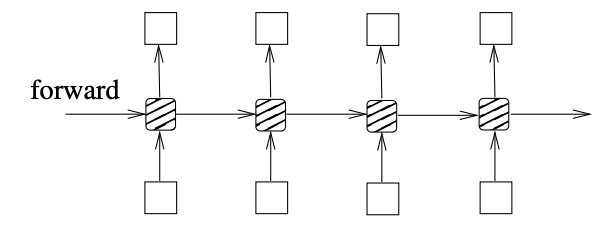


Figure 6 LSTM network

A picture containing object, clock

Description automatically generated

Figure 7 Bidirectional LSTM network

Two bidirectional LSTM units with identical architecture are used for this approach. Hence, this model is also called a Siamese Bidirectional LSTM.

# Experimental Evaluation

* 1. Dataset

This experimental evaluation uses a dataset from an experiment conducted by Mohler, Bunescu, and Mihalcea [13].

The dataset consists of data structures questions for introductory computer science assignment at the University of North Texas. There are total of 80 questions and 31 students enrolled in the course. In total, the dataset consists of total 2273 student answers since some students did not submit any answer for some questions.

Each answer is graded manually by two human graders, and the score is an integer ranging from 0 to 5, where 5 indicates a perfect answer. The average of the two scores is then used as gold standard of this experiment.

* 1. Evaluation Metrics
     1. Pearson Correlation Coefficient

Pearson correlation coefficient measures the linear correlation of two variables. The coefficient ranges from -1 to 1, where 1 means the two variables are positively correlated and -1 negatively correlated.

Pearson correlation coefficient of two variables and is defined as below.

Equation 14 Pearson correlation coefficient

Equation 15 Covariance

Equation 16 Standard deviation

Figure 9 below shows the correlation between two variables X in X-axis and Y in Y-axis.

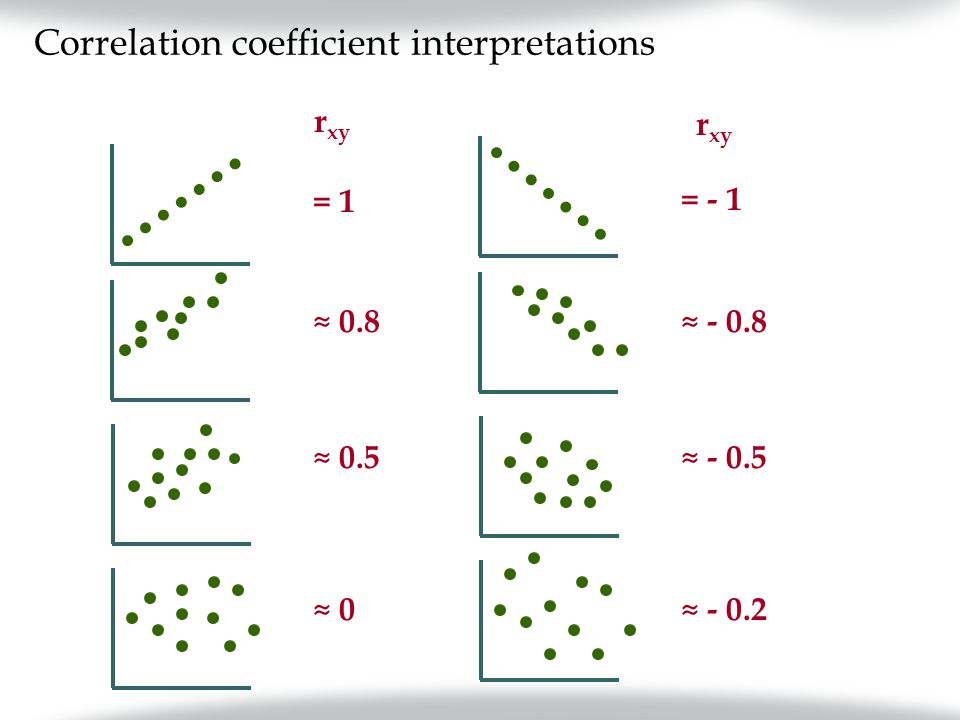


Figure 9 Pearson correlation coefficient

Pearson correlation coefficient is calculated to evaluate the approach. However, it only represents linear correlation of the two vectors and does not represent the error or how much the vector deviate from each other. For example, vectors and [ will have the coefficient equals to 1. Therefore, this metric cannot be used alone to determine the performance of a model.

* + 1. Rooted Mean Square Error

Rooted mean square (RMSE) is the square root of the arithmetic mean of the squares of the difference between the two variables.

Equation 17 RMSE

In this experiment, are target gold standard values, and are predicted values by the model. Since this metric represents the difference between the two values, this metric can be used to observe the performance of a model. Lower RMSE means the model could predict values that are closer to the given dataset.

For example, the RMSE of an error of 0.5 in grading would be 0.25, while an error of 1 in grading would result be 1. From this example, it can be concluded that even though the RMSE is 4 times as large, does not mean that it is 4 times as bad.

* + 1. Mean Absolute Error

Mean absolute error (MAE) measures the difference of two variables. It is the arithmetic mean of the absolute difference of the two variables.

Equation 18 MAE

are observed values and are predicted values. This metric also represents the difference between the two values and can be used to evaluate model performance. Lower MAE also means that the model could predict values close to the actual data.

Using similar example from the previous section, an error of 0.5 in grading would have MAE 0.5, and error of 1 would result in 1. In this case, it can be assumed that the error is twice as bad. Therefore, MAE is used to compare different models’ performance in this experiment.

* 1. Data Augmentation

The dataset only consists of 2273 tuples. This number of data is not sufficient to train a complex network, and it will cause the model to overfit the data. Hence, to avoid this issue, data augmentation is applied to obtain larger dataset.

Data augmentation is a method to increase the size and diversity of the dataset for training without actually collecting new data, so the new data is obtained from the dataset itself [20].

In this experiment, student answers that are given perfect score 5.0 by human graders are used as new reference answer. Thus, if out of students received perfect grade in a particular question, new data tuples can be generated for that question. By performing this technique, around 35000 data samples were generated and used to train the model.

* 1. Evaluation
     1. Implementation

The model was built using Keras, a high-level neural networks API that is able to run on top of TensorFlow [21]. The code is written on Python and implemented with the help of other machine learning libraries, such as pandas, a data analysis and manipulation tool [22]; NumPy, a library that supports large dimensional array operations [23]; scikit-learn, a machine learning library; and NLTK (Natural Language Toolkit), a natural language processing library.

* + - 1. pandas

pandas is a data analysis and manipulation tool [22]. This library is used in this experiment to manipulate and process the data so that it can be used to train the model.

* + - 1. NumPy

NumPy is a library that supports large dimensional array operations [23]. This library is used in this experiment since Keras requires its input to be in NumPy array format.

* + - 1. scikit-learn

scikit-learn is a machine learning library that supports tools for predictive data analysis. It provides numerous machine learning features such as classification, regression, clustering [24]. In this experiment, its model selection, metrics, and pre-processing built-in functions are used.

* + - 1. Natural Language Toolkit

NLTK is a natural language processing library [25]. It provides list of English stopwords and tokenization, which are used in this experiment.

* + 1. Model Selection

The model selection method used was a combination of three-way data splits and 5-fold cross validation methods.

The dataset was first split into training and test set on 80:20 ratio. Then, the model is trained using 5-fold cross validation of the training data, and the best model is chosen based on the average of RMSE and MAE. The fold that has lowest average of RMSE and MAE is selected. Lastly, the chosen model is used to predict values from unseen data (test set) and the result from test set is used to evaluate the overall model performance.

The data split ratio 80:20 is chosen upon some considerations. The training set, which is used to train the model, might be too small if 70:30 ratio is used, considering the complexity of the model. On the other hand, 90:10 ratio will produce very small test set. Since test set will be used to assess the model performance, if it is too small, it may not correctly represent the performance.

* + 1. Training

For each fold on training set, the model is trained on GPU using 150 epochs since the loss seems to be stable around that iteration. The parameter is updated using mini-batch stochastic gradient descent algorithm, specifically AdaGrad optimization method, and the loss function is mean absolute error (MAE) as stated in Equation 16.

AdaGrad is used because it is not sensitive to hyperparameters as it is dynamic and adaptable to the data and generate different learning rates for different features. Parameters of features that occur frequently will be applied lower learning rates than of features that occur infrequently [26]. The input of the model used in this experiment is diverse, and hence, using AdaGrad as the optimization method will help the model to train better.

* 1. Performance of Existing Model

Using the evaluation techniques explained in the previous section, the existing model performance was as stated below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Pearson coefficient** | **RMSE** | **MAE** | **Avg(RMSE+MAE)** |
| 0.4416 | 0.9815 | 0.6471 | 0.8143 |

Table 1 Performance metrics of existing approach

From the metrics above, it is shown that the model does not perform very well as the Pearson correlation coefficient is less than 0.5. The low coefficient means that the predicted values are not linearly correlated to the target values.

Most importantly, the MAE is fairly high, which is 0.6471. It means that for all gold standard score, the average difference with the predicted score is 0.6471 out of 5.0, which the error is 12.94% of the total score. For example, if the gold standard score is 4, the model would probably predict the score to be 4.6471, which is closer to 5 than to 4.

Below is the training and test error against the number of iterations.

A close up of a mans face

Description automatically generated

Figure 10 Training error of original model

A screen shot of a social media post

Description automatically generated

Figure 11 Test error of original model

As we can see from the graphs above, the graph of the test error against number of iteration seems to be very bumpy and spiky even though the graph of training error is very smooth. It means that the model does not perform well on the unseen data and it is overfitting on the training dataset. As a result, the model is unable to predict accurately on unseen data and produce high error on test dataset.

* 1. Improving Model Performance

Since the original model is overfitting on the training data, the model should be improved to result on better performance. There are many techniques that can be implemented to improve the model performance.

* + 1. Batch Normalization

The input and output of batch normalization layer are four dimensional vectors of batch, channel, and two spatial dimensions [19]. It normalises each input of a layer by subtracting mini-batch mean and dividing it by the mini-batch standard deviation. However, it may change the input’s representation [10].

For input and output over a mini-batch B, batch normalization needs to learn parameters and , where is a constant added for numerical stability.

Equation 10 Mini-batch mean

Equation 11 Mini-batch variance

Equation 12 Normalization

Equation 13 Scaling and shifting

The original model only performs batch normalization on the last perceptron layer. In this experiment, additional batch normalization is applied on each of the LSTM layers.

* + 1. Regularization

Regularization is a strategy to reduce error on unseen data or test error by altering the learning algorithm [8]. Even though this could increase the training error, reducing test error would improve the overall performance of a model since with lower test error, the model would predict more accurately on unseen data.

* + - 1. Weight regularization

Weight regularization is done by adding penalty or regularization term on the cost or loss function. The most commonly used weight regularization is L2 regularization, also known as ridge penalization. This approach uses L2 regularization term, which is defined below [27].

Equation 19 L2 regularization term

This term is added to the cost function . Since the goal of the algorithm is to minimize the cost function, when the penalization is increased, the weights would be decreased.

Equation 20 Cost function with L2 regularization

When the weights have smaller values, the model would be simpler and, as a result, less prone to overfitting.

In this experiment, different values of parameter will be used to evaluate the optimal parameter.

* + - 1. Dropout

Dropout is a method to avoid overfitting by omitting some neurons in the network.

A close up of a map

Description automatically generated

Figure 8 Dropout on neural network with 2 hidden layers

By performing dropout, random neurons, which the number of them is determined by the dropout rate, are omitted. Hence, the model becomes less complex and it can better learn important features from the dataset [11].

In the original model, the neurons in perceptron layer is dropped out 50%. Hence, this experiment will try to perform dropout on the LSTM layer and try different dropout rates on the perceptron layer.

* + 1. Question Demoting

During data pre-processing, all words in reference and student answers that are in the question are removed. However, there are some cases where the answer is in the question sentence. For example, the question “What data structure is more appropriate for scheduling printing jobs at a printer, a stack or a queue?”. The answer of that question would be either “Stack” or “Queue”. When question demoting is performed on this question and its answers, the answers will be empty. Hence, to avoid this issue, the question is altered by removing the options, i.e. “a stack or a queue”.

* + 1. Classification Model

The model is converted into a classification model with expectation that it result in better performance.

* + - 1. Motivation

The answers from the dataset is scored using ordinal values, which are 0, 1, 2, 3, 4, and 5. However, there is no rubric on how to score the answers, so each grader may have different opinion on how an answer should be graded. For example, one might think an answer should be graded 2, while the other think it is 3.

Moreover, the model that is used to predict the score is a regression model. The predicted answer would be a continuous number, and since there is no rubric on how to score the answer, there is no way to determine how correct an answer is based on the score.

* + - 1. Implementation

The model is modified into a multi-category classification model, where the output is three classes, 0, 1, and 2. An answer is labelled 0 or ‘wrong’ when it is completely wrong and completely different from the reference answer. It is labelled 1 or ‘not quite correct’ when it is only partially correct. Lastly, it is labelled 2 or ‘correct’ if it is a perfect answer and exactly the same as the reference answer.

The model is built by changing the output layer of the original model to a softmax layer with 3 outputs. Then, it is trained with the dataset, where an additional pre-processing is applied. The target output, which is the average scores of two graders , are discretized into 3 categories using the function below.

Equation 21 Target output discretization

* + - 1. Evaluation metrics

Accuracy, precision, recall, and F1 score are used to evaluate the model.

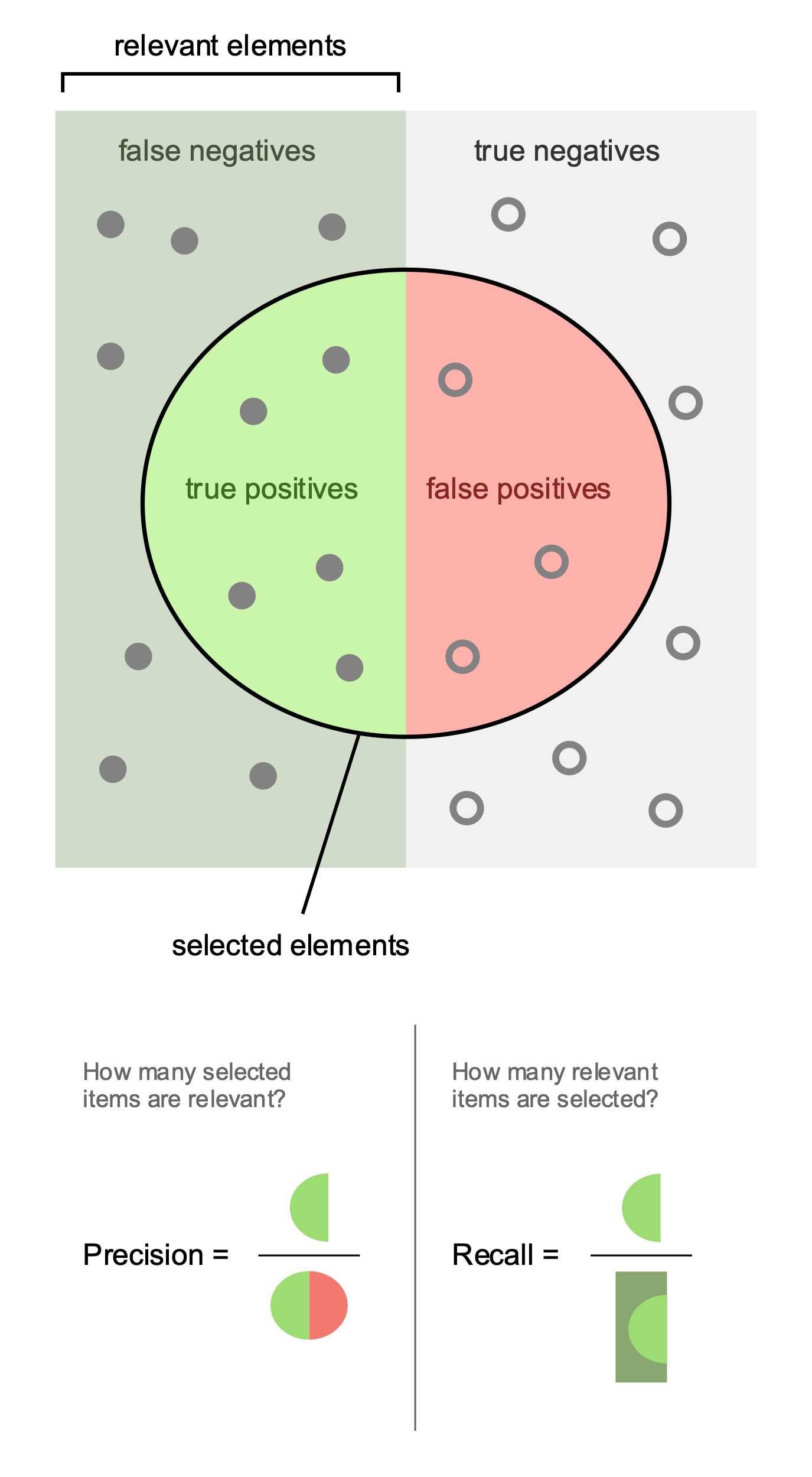


Figure 12 Classifier terms

Accuracy is the percentage of data that are correctly predicted by the model. However, accuracy is often misleading since it is imbalanced. A dataset with 95 positive data and 5 negative data will still have accuracy of 95% although it predicts all data to be positive [28].

Precision is the ratio of data of a class that is correctly predicted over the size of data that is predicted to be in that class, but it is biased by the false positive

Recall is the ratio of data of a class that is correctly predicted over the size of data that is actually in that class, and it is biased towards the false negative.

F1 score was introduced to address the disadvantages of recall, precision, and accuracy.

* 1. Experiment Results

This section will discuss the performance of the model when improvement techniques from the previous section are applied. For each technique, different values of parameter, if possible, are implemented and evaluated to obtain the best result. Note that the most optimal values are in bold. Complete results of the experiment on regression model can be found on Appendix A: Experiment Results (Regression Model).

* + 1. Recurrent Batch Normalization Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Pearson** | **RMSE** | **MAE** | **Avg(RMSE,MAE)** |
| Original | 0.4416 | 0.9815 | 0.6471 | 0.8143 |
| Recurrent BN | **0.4426** | **0.9648** | **0.6429** | **0.8039** |

Table 2 Experiment results on recurrent batch normalization

Applying batch normalization on the recurrent network improves the model performance for all metrics since it helps the model to perform better on test data.

* + 1. Regularization Results

Several experiments on regularization were conducted on different part of the model. Different values of parameter are also experimented to obtain the best result of the regularization.

#### Weight regularization on LSTM units

Experiments on different parameters of the L2 regularization is conducted. All experiments have same parameter value for bias, weights, and recurrent parameters.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Pearson** | **RMSE** | **MAE** | **Avg(RMSE,MAE)** |
| 0.01 | 0.4512 | 0.9787 | 0.6293 | 0.8040 |
| **0.001** | **0.4526** | **0.9556** | **0.6466** | **0.8011** |
| 0.0001 | 0.4513 | 0.9563 | 0.6467 | 0.8015 |

Table 3 Experiment results of weight decay on LSTM units

The result shows that parameter 0.001 has the best performance among other parameters on all metrics.

* + - 1. Weight regularization on perceptron

Weight decay was applied to the last 2 perceptron layers on the 2 multi-layer perceptron on vertical layer and perceptron layer (refer to Figure 1).

Values for bias and weights parameter are the same, as stated below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Pearson** | **RMSE** | **MAE** | **Avg(RMSE,MAE)** |
| **0.01** | 0.4471 | **0.9694** | **0.6309** | **0.8002** |
| 0.001 | 0.4491 | 0.9677 | 0.6571 | 0.8124 |
| 0.0001 | **0.4504** | 0.9952 | 0.6349 | 0.8151 |

Table 4 Experiment results of weight decay on perceptron

Weight decay with bias and weight parameter 0.01 results on model with the best performance. Even though parameter 0.0001 has the highest Pearson correlated coefficient, as elaborated on Section 4.2.1, it cannot be used independently to evaluate model performance.

* + - 1. Dropout on LSTM

Dropout with different rates are applied to the model on the LSTM layers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rate** | **Pearson** | **RMSE** | **MAE** | **Avg(RMSE,MAE)** |
| 0.2 | 0.4476 | 0.9697 | 0.6445 | 0.8071 |
| 0.4 | **0.4506** | **0.9632** | **0.6432** | **0.8032** |
| 0.5 | 0.4413 | 0.9599 | 0.6629 | 0.8114 |

Table 5 Experiment results of dropout on LSTM layers

From the experiment, it is found that dropout rate 0.4 has the best result among other rates.

* + - 1. Dropout on perceptron

On the original model, dropout rate 0.5 is used on the perceptron layers. To find out the optimal rate, first lower rate is used to see if it improves the model. Then, after seeing that lower rate does not improve the performance, higher rate is used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rate** | **Pearson** | **RMSE** | **MAE** | **Avg(RMSE,MAE)** |
| 0.2 | 0.4436 | 1.0034 | **0.6369** | 0.8202 |
| 0.5 | 0.4416 | 0.9815 | 0.6471 | 0.8143 |
| 0.6 | **0.4504** | **0.9778** | 0.6450 | **0.8114** |

Table 6 Experiment results of dropout on perceptron layers

Dropout with rate 0.6 has the best overall performance even though its MAE is still higher than model with rate 0.2.

All regularization methods are combined, and based on the experiments above, the optimal parameter values are used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Pearson** | **RMSE** | **MAE** | **Avg(RMSE,MAE)** |
| Original | 0.4416 | **0.9815** | 0.6471 | **0.8143** |
| Regularized | **0.4486** | 1.0199 | **0.6294** | 0.8247 |

Table 7 Experiment results on regularization methods

However, the performance decrease as the model seem to be underfitting the data. Hence, dropout rate is reduced back to 0.5, and afterwards, recurrent batch normalization is implemented along with all the regularization methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Pearson** | **RMSE** | **MAE** | **Avg(RMSE,MAE)** |
| Original | 0.4416 | 0.9815 | 0.6471 | **0.8143** |
| Regularization methods | **0.4494** | **0.9504** | 0.6523 | **0.8014** |
| Regularization and recurrent batch normalization | 0.4461 | 1.0000 | **0.6200** | 0.8100 |

Table 8 Experiment results on regularizations

The overall performance is better after the dropout rate is reduced. Then, recurrent batch normalization is added to the model. However, the performance does not seem to be improved.

* + 1. Question Demoting

Question demoting is applied to the dataset, and the original model is trained using the demoted answers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Pearson** | **RMSE** | **MAE** | **Avg(RMSE,MAE)** |
| Original | 0.4416 | 0.9815 | 0.6471 | 0.8143 |
| Question demoting | **0.469** | **0.9761** | **0.6204** | **0.7983** |

Table 9 Experiment results on question demoting

The performance improved quite significantly after question demoting is applied. Hence, question demoting is performed along with the optimal models from the previous section.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Pearson** | **RMSE** | **MAE** | **Avg(RMSE,MAE)** |
| Regularized | **0.4639** | **0.9604** | 0.6124 | **0.7864** |
| Regularized and BN | 0.4601 | 0.9816 | **0.6091** | 0.7954 |

Table 10 Experiment results on optimal model with question demoting

The model without recurrent batch normalization is shown to perform better as the overall metrics are better, and among other improvement, this method has the highest performance.

A close up of a piece of paper

Description automatically generated

Figure 13 Training loss of optimal model

A screenshot of a cell phone

Description automatically generated

Figure 14 Validation loss of optimal model

Compared to the original model, the performance of the model after improvement is better, especially on unseen test data. The validation loss is significantly lower and more stable than the one from the original model.

* + 1. Classification Results

Using the parameters from the previous sections, the classification model is trained using dataset that has been applied question demoting.

* + - 1. Original model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1 Score** | **Support** |
| 0 | 0.00% | 0.00% | 0.00% | 92 |
| 1 | 55.53% | 21.01% | 30.49% | 1266 |
| 2 | 80.68% | 95.68% | 87.54% | 4726 |
| **Weighted avg** | 74.23% | 78.70% | 74.35% |  |
| **Accuracy** | 78.70% | | |  |

Table 11 Experiment results on classification model

* + - 1. Original model with question demoting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1 Score** | **Support** |
| 0 | 31.13% | 51.09% | 38.68% | 92 |
| 1 | 51.12% | 36.05% | 38.32% | 1266 |
| 2 | 83.09% | 90.96% | 86.85% | 4726 |
| **Weighted avg** | 75.65% | 77.81% | 76.02% |  |
| **Accuracy** | 77.81% | | |  |

Table 12 Experiment results on classification model with question demoting

* + - 1. Model with regularization and question demoting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1 Score** | **Support** |
| 0 | 0.00% | 0.00% | 0.00% | 92 |
| 1 | 47.96% | 19.51% | 27.73% | 1266 |
| 2 | 80.97% | 95.41% | 87.60% | 4726 |
| **Weighted avg** | 72.87% | 78.17% | 73.82% |  |
| **Accuracy** | 78.17% | | |  |

Table 13 Experiment results on classification model with regularization and question demoting

Based on the results above, it can be concluded that applying regularization and question demoting does not improve the performance of the model as the accuracy is lower.

1. System Implementation
   1. Software Architectural Patterns

The system is implemented using Django framework, and its interface is built using React library.

Django is an open-source, high-level Python web framework that supports rapid development and clean, pragmatic design [29]. It follows the Model-View-Template architecture pattern, also known as MVT. The abstraction layer, model, provides structuring and manipulating the data. The view layer is responsible for the logical operations and processing user requests, including returning the responses. Lastly, template layer renders the information to be presented to the user [30].

In this system, the template layer is implemented with the support of Django REST Framework and React library. Django REST Framework, which stands for Representational State Transfer, is a powerful and flexible toolkit for building Web APIs [25]. The API helps ease the usage of Django Server as a REST API. Information from Django Server is passed through Django REST Framework to React App. React App will then provide interface that users can see.

* 1. System Interface

A prototype system implementation has been built using the architecture mentioned in the previous section. This system is built with the intention to collect more data on conceptual short-answered data structure questions, and the meantime, serving as a platform from students to practice. Data obtained from this system can be used to train the LSTM network and hopefully could improve the model performance.

* + 1. General

This section covers pages that is accessible for general use.

* + - 1. Sign up

Prior to using the system, everyone has to create an account in order to access the website. In this page, user can choose which type of user to sign up, admin or student.

A screenshot of a cell phone

Description automatically generated

Figure 15 Sign up page

* + - 1. Log in

Once the account is approved, user can sign in as the respective type of user they signed up for.

A screenshot of a cell phone

Description automatically generated

Figure 16 Log in page

* + 1. Admin

Admin users in this website will be able to upload questions and score answers. The scores will be used as gold standard during training process.

* + - 1. Questions

Admin can view all questions in the database from this page. The reference answer of the respective question is also displayed in this page.

A screenshot of a cell phone

Description automatically generated

Figure 17 Admin questions list

By clicking “Details” button on the right column of the table, user can view all answers of the respective question submitted by any student. Scores of the answer is also displayed. System score column will show the scores predicted by the neural network.

Admin can score the answers through this page by clicking “Score” button on top right. More answers for this particular question can also be added automatically by uploading a CSV file that contains answers and their scores.

A screenshot of a social media post

Description automatically generated

Figure 18 Admin question details

To upload new questions to the system, admins can upload by typing down the questions manually and the reference answer, where the questions will be added one by one, or by uploading a CSV file, where multiple questions can be added simultaneously.

A screenshot of a cell phone

Description automatically generated

Figure 19 Add questions

* + - 1. Posts

Questions uploaded will be grouped in posts. Therefore, questions can be grouped together so that they can be easier to view.

A screenshot of a cell phone

Description automatically generated

Figure 20 Admin posts list

A screenshot of a cell phone

Description automatically generated

Figure 21 Admin post details

* + - 1. Answers

Answers from all students can be viewed in this page. Their scores can also be viewed from this page. However, scoring the answer can only be done in the questions list page.

A screenshot of a social media post

Description automatically generated

Figure 22 Admin answers list

Answers can also be added automatically by uploading a CSV file containing the answers, their questions, and the scores.

A screenshot of a cell phone

Description automatically generated

Figure 23 Add answers

* + - 1. Neural network model

The neural network model is also attached to this system. The system will save the predicted scores of the last training. When new data is added, the model can be trained using the new dataset from this page. After training, all answers in the database will be predicted a new score using the new model. Lastly, the performance metrics will also be displayed in this page.

A screenshot of a cell phone

Description automatically generated

Figure 24 Model details

* + - 1. Students

Admins can view all students in the system. From this page, the students can also be approved so that they can sign in using their accounts.

A screenshot of a cell phone

Description automatically generated

Figure 25 Students list

By clicking “Details” button, user can view all answers submitted by the student and their respective questions.

A screenshot of a cell phone

Description automatically generated

Figure 26 Student details

* + 1. Student

Student is another type of user in this system. A student can only view questions and submit an answer of the questions.

* + - 1. Questions

This page shows all questions available in the system. Student can only answer each question once, and the answer cannot be changed once submitted. The score of each answer is also displayed in this page.

A screenshot of a cell phone

Description automatically generated

Figure 27 Student questions list

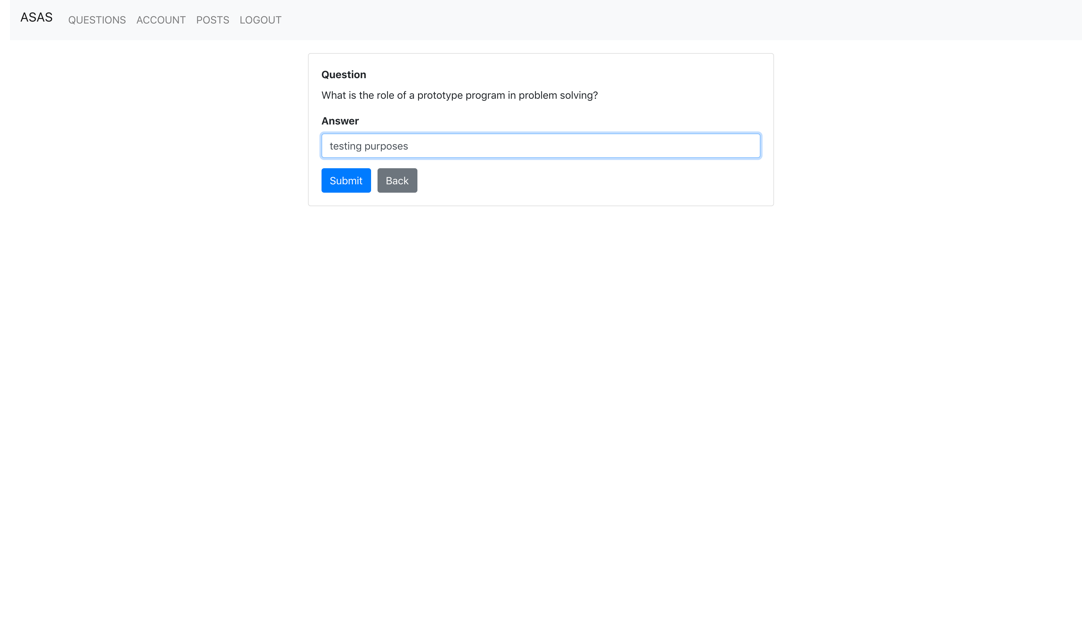


Figure 28 Student answer question page

* + - 1. Posts

Students can also view questions based on posts by clicking on the “Details” button on the rightmost column.

A screenshot of a cell phone

Description automatically generated

Figure 29 Student posts list

* + - 1. Account

Users can view and update their details from this page.

A screenshot of a social media post

Description automatically generated

Figure 30 Student account details

A screenshot of a cell phone

Description automatically generated

Figure 31 Student edit account

# Conclusion

Reference

|  |  |
| --- | --- |
| [1] | M. Zhang, “Contrasting Automated and Human Scoring of Essays,” in *R&D Connection*, vol. 21, Princeton, NJ: Research & Development Educational Testing Service, 2013, pp. 1-11. |
| [2] | C. Leacock and M. Chodorow, “C-rater: Automated Scoring of Short Answer Questions,” *Computer and Humanities,* vol. 37, pp. 389-405, 2003. |
| [3] | M. Mohler and R. Mihalcea, “Text-to-text Semantic Similarity for Automatic Short Answer Grading,” in *Proceedings of the 12th Conference of the European Chapter of the ACL*, Athens, Greece, 2009. |
| [4] | F. S. Pribadi, T. B. Adji, A. E. Permanasari, A. Mulwinda and A. B. Utomo, “Automatic Short Answer Scoring Using Words Overlapping Methods,” in *Engineering International Conference*, 2016. |
| [5] | S. Hassan, A. A. Fahmy and M. El-Ramly, “Automatic Short Answer Scoring based on Paragraph Embeddings,” *International Journal of Advanced Computer Science and Applications,* vol. 9, no. 10, pp. 397-402, 2018. |
| [6] | C. Olah, “Understanding LSTM Networks,” 27 August 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 10 March 2020]. |
| [7] | S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997. |
| [8] | I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016. |
| [9] | S. Santurkar, D. Tsipras, A. Ilyas and A. Madry, “How Does Batch Normalization Help Optimization?,” 2018. |
| [10] | S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” 2015. |
| [11] | G. E. Dahl, T. N. Sainath and G. E. Hinton, “Improving Deep Neural Networks for LVCSR Using Rectified Linear Units and Dropout,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013. |
| [12] | G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, “Improving Neural Networks by Preventing Co-adaptation of Feature Detectors,” 2012. |
| [13] | M. Mohler, R. Bunescu and R. Mihalcea, “Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, Protland, Oregon, 2011. |
| [14] | Y. Goldberg and O. Levy, “word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method,” 2014. |
| [15] | T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” *Advances in Neural Information Processing Systems 26,* 2013. |
| [16] | C. D. Manning, P. Raghavan and H. Schütze, “Tokenizer,” in *Introduction to Information Retrieval*, Cambridge University Press, 2008. |
| [17] | A. Casari and A. Zheng, Feature Engineering for Machine Learning, Sebastopol: O’Reilly Media, Inc., 2018. |
| [18] | Z. Huang, W. Xu and K. Yu, “Bidirectional LSTM-CRF Models for Sequence Tagging,” 2015. |
| [19] | J. Bjorck, C. Gomes, B. Selamn and K. Q. Weinberger, “Understanding Batch Normalization,” 2018. |
| [20] | D. Ho, E. Liang and R. Liaw, “1000x Faster Data Augmentation,” Berkeley Artificial IntelligentceResearch, 7 June 2019. [Online]. Available: https://bair.berkeley.edu/blog/2019/06/07/data\_aug/. [Accessed 11 March 2020]. |
| [21] | “Keras Documentation,” [Online]. Available: https://keras.io/. [Accessed 19 March 2020]. |
| [22] | “pandas - Python Data Analysis Library,” [Online]. Available: https://pandas.pydata.org/. [Accessed 19 March 2020]. |
| [23] | “NumPy,” [Online]. Available: https://numpy.org/. [Accessed 19 March 2020]. |
| [24] | “scikit-learn: machine learning in Python,” [Online]. Available: https://scikit-learn.org/stable/. [Accessed 19 March 2020]. |
| [25] | “Natural Lanugage Toolkit,” [Online]. Available: https://www.nltk.org/. [Accessed 19 March 2020]. |
| [26] | J. Perla, “Notes on AdaGrad,” 2014. |
| [27] | “Regularization for Simplicity: L₂ Regularization,” Machine Learning Crash Course, 10 February 2020. [Online]. Available: https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization. [Accessed 12 March 2020]. |
| [28] | “Precision and Recall,” Precision and Recall, [Online]. Available: https://en.wikipedia.org/wiki/Precision\_and\_recall. [Accessed 20 March 2020]. |
| [29] | “Django,” [Online]. Available: https://www.djangoproject.com/. [Accessed 14 March 2020]. |
| [30] | “Django Documentation,” [Online]. Available: https://docs.djangoproject.com/en/3.0/. [Accessed 14 March 2020]. |
| [31] | M. A. Sultan, C. Salazar and T. Sumner, “Fast and Easy Short Answer Grading with High Accuracy,” in *Proceedings of NAACL-HLT*, San Diego, California, 2016. |
| [32] | “Django REST Framework,” [Online]. Available: https://www.django-rest-framework.org/. [Accessed 14 March 2020]. |

Appendix A: Experiment Results (Regression Model)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Remarks** | **Reg LSTM** | **Reg dense** | **Dense dropout** | **LSTM dropout** | **Batchnorm** | **LSTM batchnorm** | **Pearson** | **RMSE** | **MAE** | **Avg** |
| Original code | 0 | 0 | 0.5 | 0 | Yes | No | 0.4416 | 0.9815 | 0.6471 | 0.8143 |
| Regularize LSTM | 0.01 | 0 | 0.5 | 0 | Yes | No | 0.4512 | 0.9787 | 0.6293 | 0.8040 |
| Regularize LSTM | **0.001** | 0 | 0.5 | 0 | Yes | No | 0.4526 | 0.9556 | 0.6466 | 0.8011 |
| Regularize LSTM | 0.0001 | 0 | 0.5 | 0 | Yes | No | 0.4513 | 0.9563 | 0.6467 | 0.8015 |
| Regularize last 2 dense | 0 | **0.01** | 0.5 | 0 | Yes | No | 0.4471 | 0.9694 | 0.6309 | 0.8002 |
| Regularize last 2 dense | 0 | 0.001 | 0.5 | 0 | Yes | No | 0.4491 | 0.9677 | 0.6571 | 0.8124 |
| Regularize last 2 dense | 0 | 0.0001 | 0.5 | 0 | Yes | No | 0.4504 | 0.9952 | 0.6349 | 0.8151 |
| Dense dropout | 0 | 0 | 0.2 | 0 | Yes | No | 0.4436 | 1.0034 | 0.6369 | 0.8202 |
| Dense dropout | 0 | 0 | **0.6** | 0 | Yes | No | 0.4504 | 0.9778 | 0.6450 | 0.8114 |
| LSTM dropout | 0 | 0 | 0.5 | 0.2 | Yes | No | 0.4476 | 0.9697 | 0.6445 | 0.8071 |
| LSTM dropout | 0 | 0 | 0.5 | **0.4** | Yes | No | 0.4506 | 0.9632 | 0.6432 | 0.8032 |
| LSTM dropout | 0 | 0 | 0.5 | 0.5 | Yes | No | 0.4413 | 0.9599 | 0.6629 | 0.8114 |
| Batch normalization | 0 | 0 | 0.5 | 0 | No | No | 0.4142 | 0.9976 | 0.6582 | 0.8279 |
| Recurrent batchnorm | 0 | 0 | 0.5 | 0 | Yes | Yes | 0.4426 | 0.9648 | 0.6429 | 0.8039 |
| Regularize | 0.001 | 0.01 | 0.5 | 0 |  | No | 0.4447 | 0.9732 | 0.6481 | 0.8107 |
| Dropout | 0 | 0 | 0.6 | 0.4 | Yes | No | 0.4473 | 0.9845 | 0.6359 | 0.8102 |
| Regularize & dropout | 0.001 | 0.01 | 0.6 | 0.4 | Yes | No | 0.4486 | 1.0199 | 0.6294 | 0.8247 |
| Regularize & dropout LSTM | 0.001 | 0.01 | 0.5 | 0.4 | Yes | No | 0.4494 | 0.9504 | 0.6523 | 0.8014 |
| Reg, dropout LSTM, batchnorm | 0.001 | 0.01 | 0.5 | 0.4 | Yes | Yes | 0.4461 | 1.0000 | 0.6200 | 0.8100 |
| Question demoting | 0 | 0 | 0.5 | 0 | Yes | No | 0.469 | 0.97615 | 0.62045 | 0.7983 |
| Question demoting, optimal | 0.001 | 0.01 | 0.5 | 0.4 | Yes | No | 0.4639 | 0.9604 | 0.6124 | 0.7864 |
| Question demoting, optimal & batchnorm | 0.001 | 0.01 | 0.5 | 0.4 | Yes | Yes | 0.4601 | 0.9816 | 0.6091 | 0.7954 |