10-28-2021

Automatic Short Answer Grading Using Deep Learning

Jinzhu Luo
Illinois State University, jinzhuluo7576@gmail.com

AUTOMATIC SHORT ANSWER GRADING USING DEEP LEARNING

JINZHU LUO

37 Pages

Under the influence of the COVID-19 pandemic, traditional in-person teaching has undergone significant changes. Online courses become an essential education method. However, online teaching lacks adequate evaluation approaches. That's why exams are still indispensable. However, grading short answer exam questions can be an onerous task. In this work, we propose a novel Automatic Short Answer Grading (ASAG) model based on the Sentence BERT model. On the Short Answer Scoring V2.0 dataset, our proposed model shows improvements on accuracy, Marco F1 score, and Weighted F1 score comparing to the results obtained from the BERT model. In addition, we also compare different task functions and different lengths of answers to further evaluate our model's performance. A better result is achieved when using the regression task function. At the same time, we find that shorter answers' result is better than the result obtained from longer answers.

KEYWORDS: Automatic Short Answer Grading, Deep Learning, BERT, Sentence BERT, Transformer

AUTOMATIC SHORT ANSWER GRADING USING DEEP LEARNING

JINZHU LUO

A Thesis Submitted in Partial
Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

School of Information Technology

ILLINOIS STATE UNIVERSITY

2021

AUTOMATIC SHORT ANSWER GRADING USING DEEP LEARNING

JINZHU LUO

CONTENTS

	Page
TABLES	iii
FIGURES	iv
CHAPTER I: INTRODUCTION	1
CHAPTER II: RELATED WORK	3
CHAPTER III: METHOD	5
Recurrent Neural Network	5
Transformer	6
Self-Attention	8
BERT	10
Masked LM	12
Next Sentence Predictions (NSP)	13
Fine-Tuning	13
Sentence-BERT	17
Pooling Strategy	18
Task Function	18
Classification Task Function	19
Regression Task Function	21
CHAPTER IV: EXPERIMENT	23
Dataset	23
Experimental Setup	25
Evaluation Measures	27

CHAPTER V: RESULT AND ANALYSIS 29

CHAPTER VI: DISCUSSION 31

REFERENCES 32

APPENDIX: PARTIAL EXPERIMENTAL RESULTS 36

TABLES

Table	Page
1. SBERT Result in NLI And STSb	18
2. The difference of classification and regression	19
3. Example of Short Answer Scoring V2.0 dataset	23
4. Example of shorter answer set and longer answer set	25
5. Prediction type	27
6. Result of different task functions	29
7. Result of SBERT and BERT	29
8. Result of shorter answer set and longer answer set	30

FIGURES

Figure	Page
1. Structure for a 3-layer RNN network	5
2. Transformer model on language translation	7
3. Structure of transformer	7
4. The structure of encoder and decoder	8
5. An example of Self-Attention	8
6. The process of calculating the Query matrix, Key matrix, and Value matrix	9
7. The architecture of BERT base and BERT large	11
8. Example of Masked LM and NSP	13
9. Pair sentence classification task	14
10. Single sentence classification task	15
11. Single sentence tagging task	15
12. An example of IOB tagging	16
13. Question answering task	17
14. The process of SBERT	17
15. SBERT architecture with classification task function	20
16. SBERT architecture with regression task function	21
17. Angle between two vectors A and B	22
18. The result of different Epochs	26

CHAPTER I: INTRODUCTION

In this work, we propose a novel ASAG model based on the Sentence BERT model. Traditional automatic short answer grading involves the instructor scoring students' answers one by one for grading. This model consumes a lot of time and energy from teachers, and teachers cannot grade all the students' answers in a short time, which also causes students cannot to get timely feedback. Compared to the traditional grading model, computer-assisted assessment approaches are faster and more accurate. There are two major assessment approaches to date. The first one scores an essay based on pre-defined criteria, like grammar, diction, and coherence. It is mostly used for scoring long essays. The second approach is to compare the similarity between a reference answer and student's answer to score. In this work, we propose an ASAG system on the basis of the second approach.

Our proposed model for short-term answer scoring is based on state-of-the-art Natural Language Processing (NLP) techniques. Pre-trained transformer models have been widely adopted in solving NLP-related tasks. The BERT model is the most impressive one and records the best scores in 11 different NLP tests, including pushing the General Language Understanding Evaluation (GLUE) (Wang et al, 2018) benchmark to 80.4% (7.6% absolute improvement) and MultiNLI (Williams et al, 2018) accuracy to 86.7% (5.6% absolute improvement). Sentence BERT(SBERT) is a modification of the pre-trained BERT network that uses Siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine similarity. On the Semantic Textual Similarity (STS) Benchmark, Sentence BERT obtains an accuracy up to 88.77% on Spearman Correlation.

In our work, we compare Sentence BERT with the BERT model on the 2440 student answers' dataset, in terms of accuracy and F1 scores to determine which model is more suit for the ASAG domain. Additionally, we also explore the effect of different task functions and the effect of short answer question length on the result.

Therefore, we will first present an overview of the automatic short answer scoring system, the BERT and Sentence BERT model architectures, and then present details of our experiments.

CHAPTER II: RELATED WORK

Page (1966) is the first one who study of the computers to assisted evaluate student answer with natural language processing methods. This technology is known as Project Essay Grade. With the development of Machine learning, Pulman et al (2005) compares several machines learning techniques, including inductive logic programming, decision tree learning, and Bayesian learning, to the earlier automatic scoring approach with encouraging results.

Gütl (2007) build a e-Examiner system applied natural language pre-processing with statistic ROUGE metrics (C. Y. Lin, 2004) and statistic measures of text similarity to show a feasibility of computer-assisted and computer-based knowledge assessment systems. Bailey (2008) uses the k-neighbor classifier on a learner corpus they collected from learners completing the exercises assigned in a ESL class. Their result successful prove that viability of using NLP techniques for grade score between teacher and student responses. Madnani et al (2013) introduced eight features in ASAG, such as Bi-lingual Evaluation Understudy (BLEU) (Papineni et al,2002), and then combined all the features in a logistic regression classifier to output a prediction result. Burrows (2015) divided these studies on ASAG into several categories. It includes concept mapping, information extraction, corpus-based studies, and machine learning-based methods.

With the rapid development of NLP, the idea of automatic evaluation of open-ended question responses has attracted many researchers. And with it comes a large number of competitions. The most famous of these are the Automated Student Assessment Prize and the SemEval'13 task.7 In the Automated Student Assessment Prize competition, the Quadratic Weighted Kappa (QWK) method was used as a success criterion. SemEval'13 Task 7 was the first large-scale, non-commercial ASAG competition. And those competitions promote the development of deep learning and ASAG.

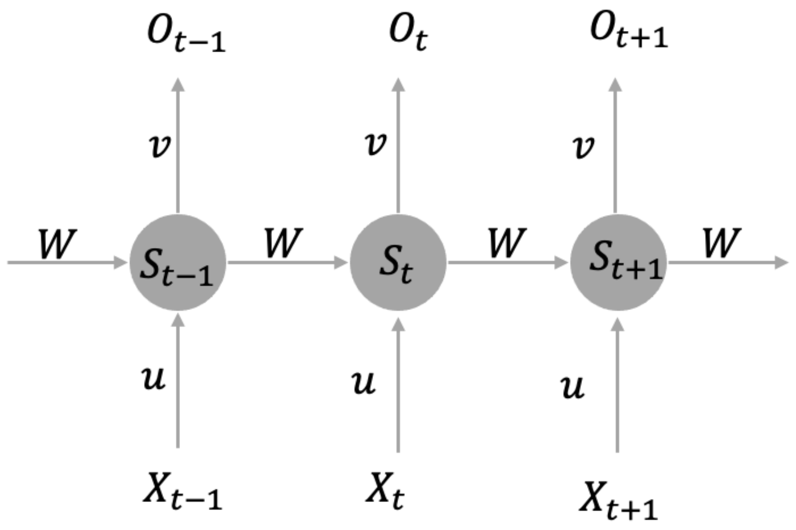
Recently, several pretrained models such as BERT (Devlin et al, 2018), GPT-2(Radford et al, 2019), ELMo (Peters et al, 2018) have been produced as a result of the processing of large corpora with deep learning techniques. These language models are pre-trained and made publicly available. They are also can use for different NLP downstream tasks with prior fine-tuning. Sung et al (2019) have shown that by updating the pre-trained BERT language model with domain-specific books and question-answer data, better results can be achieved instead of fine-tuning the model. Sentence BERT (Reimers et al, 2019) is a modification of the pre-trained BERT network that uses Siamese and triplet network structures. And SBERT also trains on the sentence similarity task dataset such as Semantic Textual Similarity dataset (Cer et al, 2016).

CHAPTER III: METHOD

Recurrent Neural Network

A Recurrent Neural Network (RNN) is a model of sequences with memory cells that can remember historical information. And we can handle contextual semantics with these memory cells. Based on this, we can use those memory cells to learn the surrounding window information and thus determine the semantics of the central word. The original RNNs were difficult to train due to vanishing gradient and gradient explosion. To solve this problem, some models such as long short-term memory (LSTM) and gated recurrent unit (GRU) were introduced. The RNNs could be used for some basic tasks, such as classification, regression, and sequence annotation.

Figure 1 Structure for a 3-layer RNN network



The figure 1 shows a complete 3-layer RNN network. X_t is the input at time step t. Usually, X_t represents the one-hot vector of the t-th word of the input sentence. S_t is the hidden state at time step t. It is calculated based on the previous hidden state and the input at the current step. S_t can captures information about what happened in all the previous time steps. The equation of S_t:

$$S_t = f(UX_t + WS_{t-1})$$

The function f usually is a nonlinearity such as tanh (Wolfram Language, 1988) or ReLU (Agarap, 2018).

O_t is the output at step t. The equation of O_t:

$$O_t = softmax(VS_t)$$

RNN uses the three same parameters (U, V, W) across all steps. This reduces the number of parameters we need to learn.

As briefly mentioned above, S_t can store previous information. But in real practice, S_t is usually unable to capture information from too many time steps ago. It can still cause information loss. Also, we can see that RNN is a sequential computational structure, which means that it cannot handle large data sets in parallel.

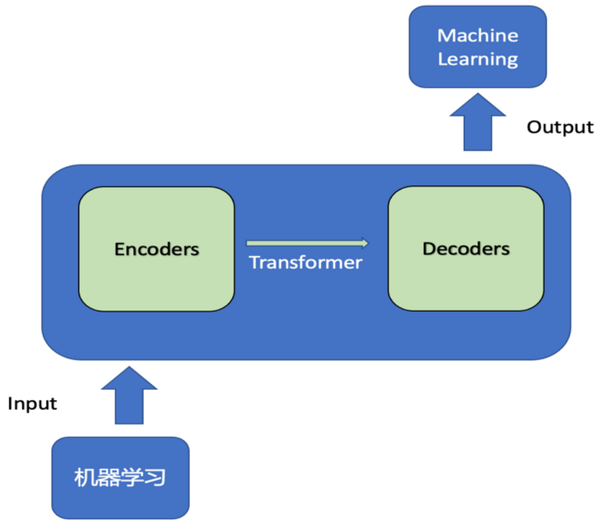
The transformer model solves the above two problems. First transformer uses the Attention mechanism. It can reduce the distance between any two positions in the sequence to a constant number, which will not cause information loss. Secondly it is not a sequential structure similar to RNN, so it has better parallelism.

Transformer

Transformer model (Vaswani et al, 2017) is a deep learning model that employs the attention mechanism to accelerate model training. It is also appropriate for parallelized computation. It also performs better than classical deep learning models such as RNN.

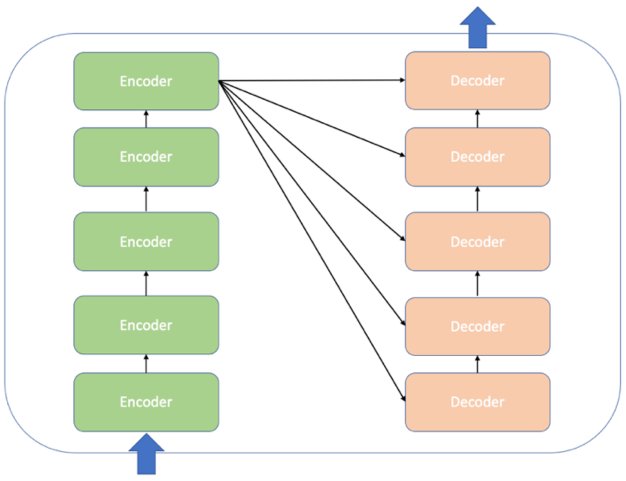
Transformer is made up of two parts: Encoder model and Decoder model. When we enter a text, the text data is first process by an Encoder model, and then the encoded data is passed to a module called Decoder to be decoded.

Figure 2 Transformer model on language translation



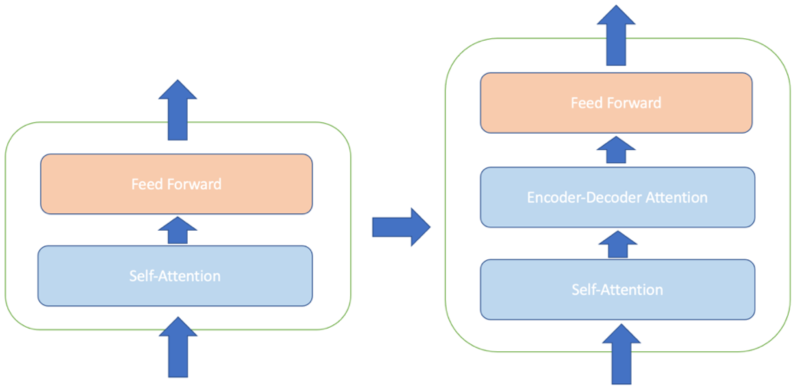
Encoder model is created by stacking many encoders with the same structure. Each encoder's output is the input of the next encoder layer, and the input of the bottom encoder layer is the original input. Decoder model is similar, but the output of the last encoder layer is the input to each encoder layer, as required by the attention mechanism.

Figure 3 Structure of transformer



Each layer's encoder is the same structure, consisting of a Self-Attention layer and a feed-forward network (fully connected network). The decoder of each layer has the same structure; it has a general Attention layer in addition to the Self-Attention layer and the fully connected layer, and this Attention layer forces all decoders to consider the output of the last encoder layer.

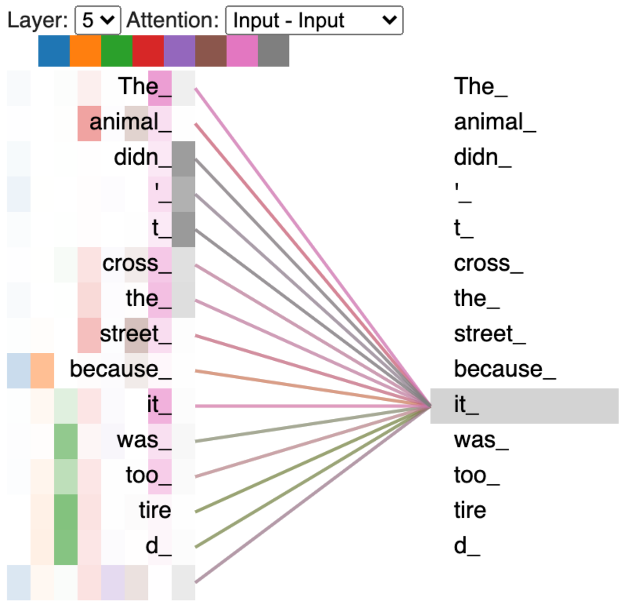
Figure 4 The structure of encoder and decoder



Self-Attention

Self-Attention is the core of transformer. And it is the calculation of the attention of each word within the same sentence and which words within the same sentence should be focused on. Simply, Self-Attention is learning a weight for each word of the input vector.

Figure 5 An example of Self-Attention


Layer: 5 Attention: Input - Input

In the transformers Encoder model, the input first goes through the self-attention module to obtain the feature vector:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

In the following, we will introduce the attention formula step by step:

1. We first transform the input statement into an embedding matrix x.

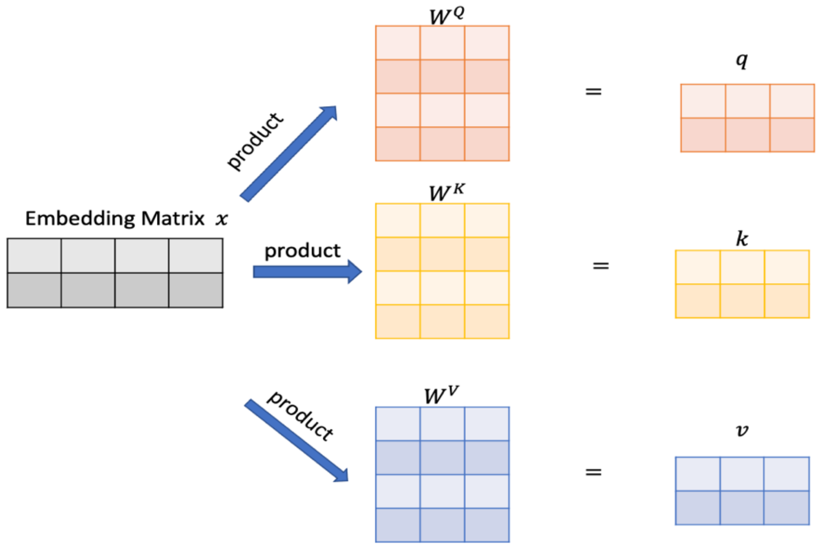
2. Calculate the Query matrix (q), Key matrix (k) and Value matrix (v). They are obtained by multiplying the embedding matrix x by three different weight matrices W^Q, W^K and W^V, which also have the same dimensions.

$$q = x * W^Q$$

$$k = x * W^K$$

$$v = x * W^v$$

Figure 6 The process of calculating the Query matrix, Key matrix, and Value matrix



The idea of three matrixes q, k, v is from the field of information retrieval. q is the query. k is the key. v is the value. And (k, v) is the key-value pair that uses query keywords to find the most relevant search results.

3. Calculate a score for each matrix.

$$Score = q * k^T$$

4. In higher dimensions, to prevent the vanishing gradient problem when the softmax function back propagates due to the large value of QK^T, we divide $Score$ by $\sqrt{d_k}$.
5. Apply a softmax activation function to the score.

$$softmax(\frac{Score}{\sqrt{d_k}})$$

6. To obtain the weighted score of each input matrix.

$$softmax(\frac{Score}{\sqrt{d_k}})v$$

The structure of Decoder model is shown in Figure 4, which differs from Encoder model in that Decoder model has one more Encoder-Decoder Attention, and two Attention layers are used to calculate the weights of input and output respectively.

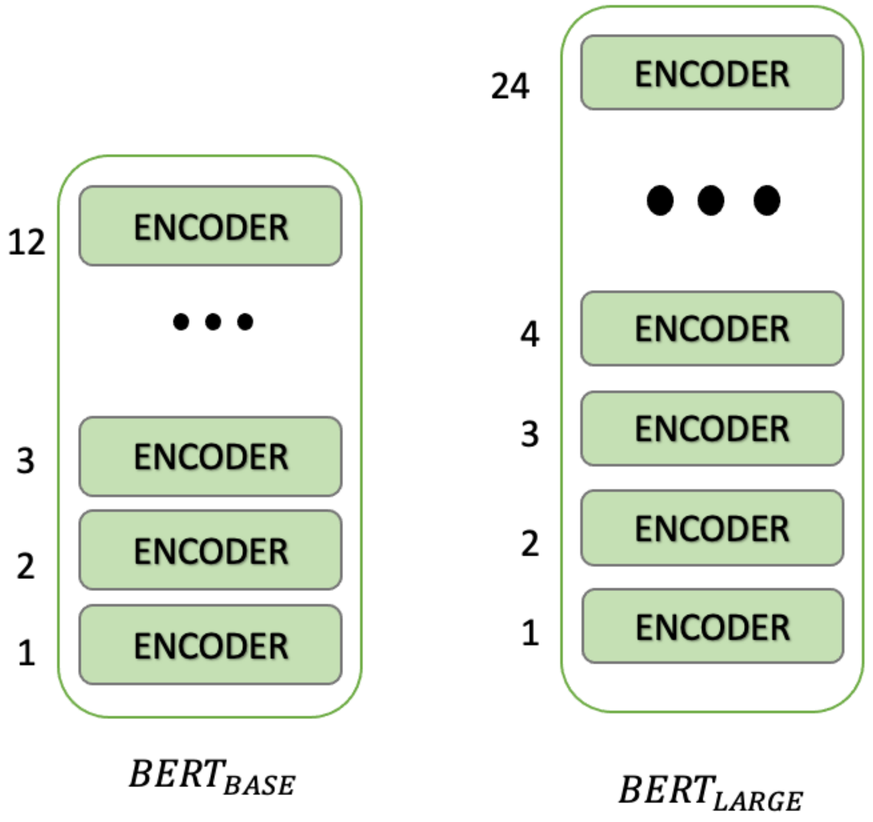
1. Self-Attention: the relationship between the current translation and the previous text that has been translated.
2. Encoder-Decoder Attention: the relationship between the current translation and the encoded feature vector.

BERT

BERT, which is a bi-directionally trained language model. This provides us now a much deeper understanding of the linguistic context compared to one-way language models.

BERT uses a novel technique called Masked Language Model (LM): it randomly masks words in a sentence and then tries to predict them. This means that the model looks in both left-to-right and right-to-left directions, and it uses the entire context of the sentence to predict the masked words. Unlike previous language models, it considers both the previous and next tokens.

Figure 7 The architecture of BERT base and BERT large



The core of BERT is the transformer Encoder model that reads the text input. There are currently two versions of BERT:

1. BERT Base: 12-layer transformer, 12 attention heads and 110 million parameters
2. BERT Large: 24-layer transformer, 16 attention heads and 340 million parameters

The input to the BERT encoder is a series of tokens that is first converted to vectors and then processed in the neural network. Each of these inputs is a combination of three embeddings.

- Token Embedding: the [CLS] token is added to the input word tokens at the beginning of the first sentence and the [SEP] token is inserted at the end of each sentence.
- Segment Embedding: A token indicates either sentence A or sentence B is added to each token. This allows the encoder to distinguish sentences, assisting the model in differentiate them.
- Positional Embedding: Positional embedding is added to each token to indicate its position in the sentence. It adds positional embedding to overcome the limitations of the transformer, which, unlike RNN, does not capture "sequence" or "order" information

Masked LM

Randomly mask the words in the input sentence with the [MASK] tokens. Then we will run the entire input sequence through a BERT attention-based encoder. Finally we will calculate the [MASK] tokens by the other non-masked words in the sequence. This is the key to the masking language model. The authors of BERT also propose some operation to further improve this technique:

- To prevent the model from focusing too much on a specific location or masked token, the researchers masked 15% of the words at random.
- Masked words are not always replaced by the mask token [mask], because the [mask] token does not appear during tuning.

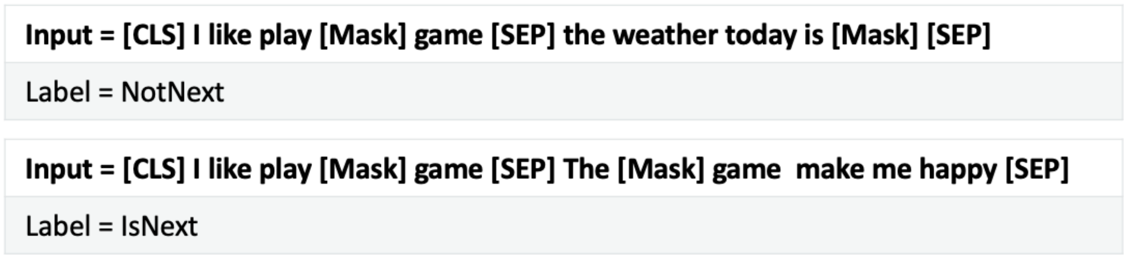
  Therefore, the researchers used the following method:
- 80% of the tokens were replaced with masked tokens [masks]

- 10% of the tokens were replaced with a randomly token
- 10% of the tokens are kept unchanged

Next Sentence Predictions (NSP)

BERT can predict whether two sentences are related or not. This is a kind of multi-Task learning. It uses the data from BookCorpus (Zhu et al, 2015). BookCorpus is a dataset composed of many books and Wikipedia. And in these datasets, the sentences before and after each book are related. For this task, BERT will extract the related sentences with 50% probability and another two unrelated sentences with 50% probability randomly, and then let the BERT model to determine whether these two sentences are related or not.

Figure 8 Example of Masked LM and NSP



Input = [CLS] I like play [Mask] game [SEP] the weather today is [Mask] [SEP]

Label = NotNext

Input = [CLS] I like play [Mask] game [SEP] The [Mask] game make me happy [SEP]

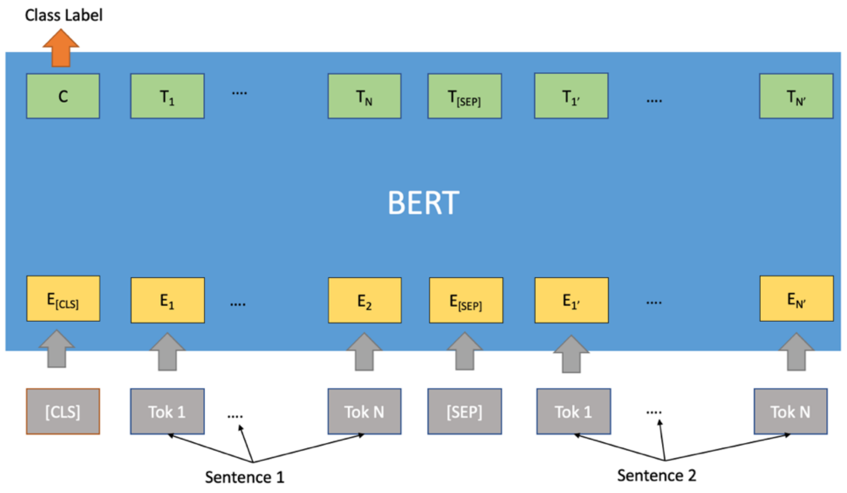
Label = IsNext

Fine-Tuning

Pre-training models such as BERT have already migrated knowledge from the corpus into the Embedding of the pre-training model through unsupervised learning of a large corpus, so we only need to add or change structures to fine-tune it for the different NLP downstream tasks. And BERT drives the latest technical results for 11 natural language processing tasks, and these 11 NLP tasks can be categorized into four major natural language processing downstream tasks.

1.For the classification task, the input is a sequence, as shown in Figure 9, where all tokens belong to the same segment (Id=0), while two sentences need to be connected with [SEP] and input to the model. The output of the last layer is connected to softmax for classification using the first special Token [CLS], and the classified data is used for fine-tuning.

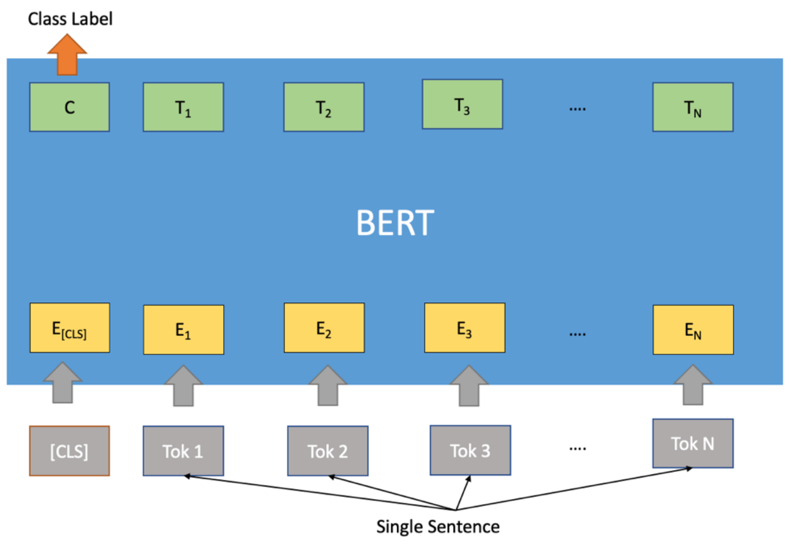
Figure 9 Pair sentence classification task



The author fine-tuning BERT model on those datasets: Multi-Genre Natural Language Inference (Williams et al, 2017), Quora Question Pairs(Chen et al, 2018), Question Natural Language Inference(Wang et al, 2018), Microsoft Research Paraphrase Corpus(Dolan et al, 2005), Recognizing Textual Entailment(Bentivogli et al, 2009) and Situations With Adversarial Generations(Zellers et al, 2018).

2.The single-sentence classification task allows a simple classification layer to be added directly to the pre-trained model, and then all parameters can be fine-tuned together on the downstream task.

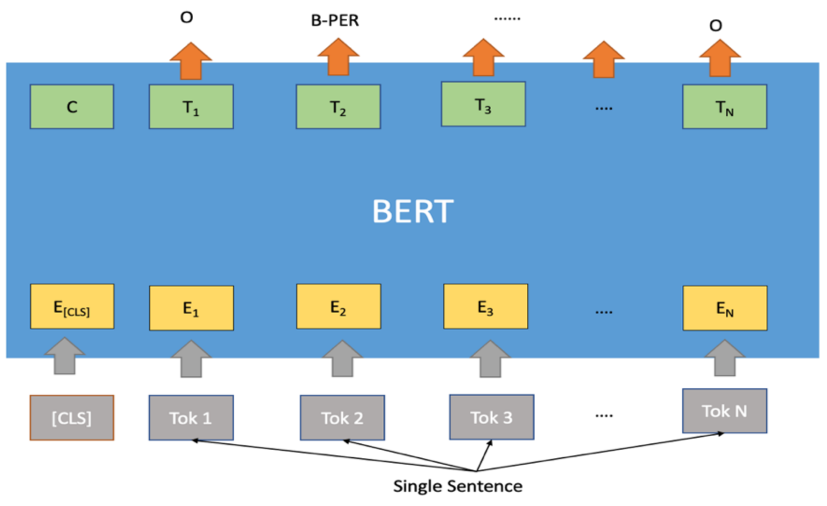
Figure 10 Single sentence classification task



The author fine-tuning BERT model on those datasets: Stanford Sentiment Treebank (Socher et al, 2013) and Corpus of Linguistic Acceptability (Warstadt et al, 2018).

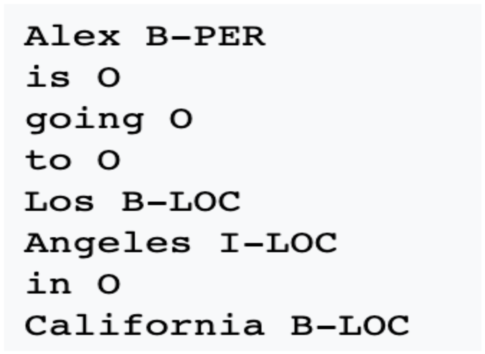
3.The single-sentence annotation task, also called Named Entity Recognition (NER), refers to the recognition of entities with specific meanings in text, mainly including names of people, places, institutions, proper names, etc.

Figure 11 Single sentence tagging task



And the author fine-tuning BERT model on CoNLL-2003 (Sang et al, 2018) dataset. Before that, we need to understand the format of the dataset. Inside–outside–beginning (IOB) tagging (Ramshaw& Marcus,1999) is a way to sequentially tag the units in a given sentence and is used to extract meaningful phrases consisting of consecutive words/word chunks from a given sentence. For a given sentence, each word is tagged as one of B (Beginning, indicating the beginning of a phrase), I (Inside, indicating the inside of a phrase), and O (Outside, indicating not in a phrase).

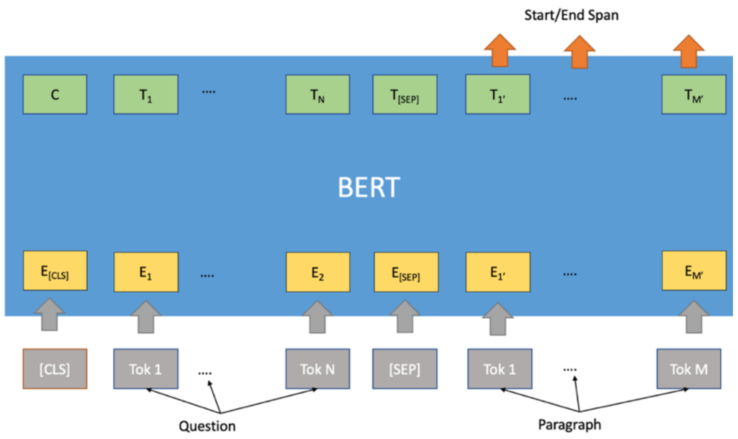
Figure 12 An example of IOB tagging



```
Alex B-PER
is O
going O
to O
Los B-LOC
Angeles I-LOC
in O
California B-LOC
```

4. The question-and-answer task first requires the questions and text to be connected with [SEP] and input to the model. Then, we input the final layer of BERT vectors to the output layer. Finally, we draw the beginning probability vector $start$ and the end of the answer probability vector end respectively. The total $loss$ is calculated based on the difference between the two and the real answer pair (start, end).

Figure 13 Question answering task

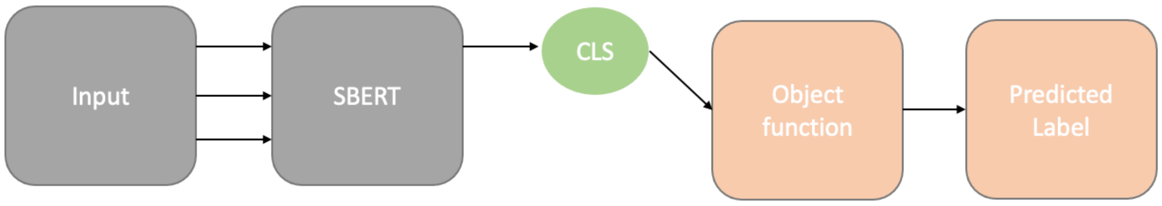


And the author fine-tuning BERT model on Stanford Question Answering Dataset (Rajpurkar et al, 2016).

Sentence-BERT

The BERT model has made great breakthroughs in all major NLP tasks. For the semantic textual similarity computation task, BERT model specifies that two sentences need to be entered into the model at the same time for information interaction. This operation causes a large amount of computational overhead. For example, there are 10,000 sentences and we want to find the most similar sentence pair, we need to compute (10,000*9999/2) times. This makes BERT unsuitable for both semantic similarity search and unsupervised tasks.

Figure 14 The process of SBERT



Sentence BERT network structure which uses twin and triplet network structures to generate semantically meaningful sentence embedding vectors. Sentences with similar semantics have a closer distance in embedding vectors. Thus, it can be used for similarity calculation such as cosine similarity. For the same 10,000 sentences, finding out the most similar sentence pair only requires 10,000 calculations.

Pooling Strategy

SBERT adds a pooling operation to the output of BERT, generating a fixed-size sentence embedding vector. Three pooling strategies are adopted in the experiments for comparison:

- CLS: directly use the output vector of [CLS] to represent the vector representation of the whole sentence
- MEAN: calculate the average of each token output vector to represent the sentence vector
- MAX: take the maximum value of all output vectors in each dimension to represent the sentence vector

Table 1 SBERT Result in NLI And STSb

Pooling Strategy	NLI	STSb
MEAN	80.78	87.44
MAX	79.07	69.92
CLS	79.8	86.62

Task Function

In deep learning, we often divide into different tasks based on the characteristics of the dataset. In our work, we use the regression method and classification method. Regression methods are supervised learning algorithms for predicting and modeling numerical continuous random variables. Use cases typically include continuously varying cases such as stock movements. A classification method is a supervised learning algorithm for modeling or predicting discrete random variables. Use cases include tasks where the output is a category, such as emails filtering and financial fraud. Classification algorithms are typically applied to predict a category (or probability of a category) rather than continuous values.

Table 2 The difference of classification and regression

	Classification	Regression
Type of output	Discrete data	Continuous data
Target	Find the decision boundary	Find the line of best fit

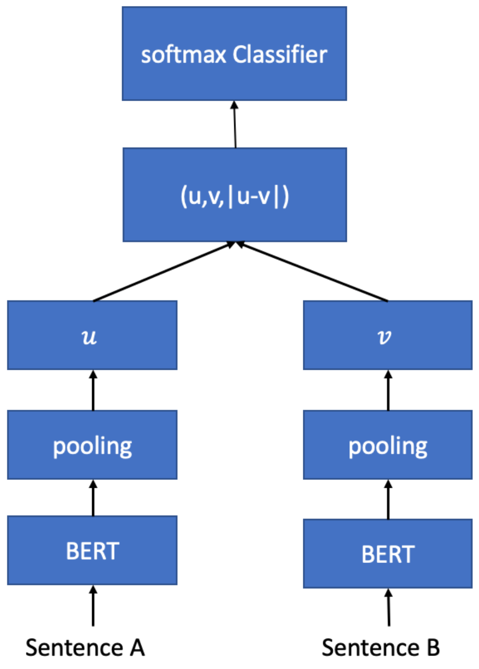
Classification Task Function

SBERT concatenate the sentence embeddings u and v with the element-wise difference $|u - v|$ and multiply it with the trainable weight $Wt \in R^{3n \times k}$:

$$o = softmax(Wt(u, v, |u - v|))$$

where n is the dimension of the sentence embeddings and k denotes the number of labels. Then SBERT optimizes the cross-entropy loss.

Figure 15 SBERT architecture with classification task function



Softmax function is used in the multi-classification process, which maps the output of multiple neurons into the (0,1) interval. So, the softmax function can give the probability distribution of an input being classified into each category, and the cumulative sum of these probability values is 1. Suppose we have an array E and e_i denotes the i-th element probability value in E. Then the softmax value S_i of this element is

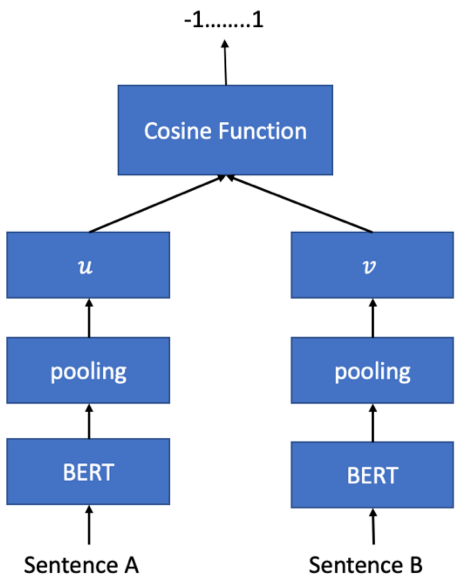
$$S_i = \frac{e^i}{\Sigma_j e^j}$$

With the predicted label S_i and the correct label, you can easily measure the loss of the model prediction by characterizing the similarity between the two labels. The cross-entropy loss is a way to characterize the similarity of the two probabilities.

Regression Task Function

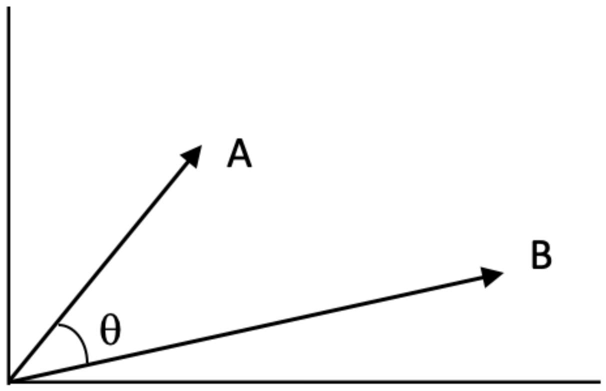
The cosine similarity between the two sentence embeddings u and v is computed. And the result is compared to the real label. SBERT uses mean squared-error loss as the objective function.

Figure 16 SBERT architecture with regression task function



Cosine similarity measures the similarity between two vectors by measuring the cosine of the angle between them. The cosine of an angle of degree 0 is 1, while the cosine of any other angle is not greater than 1. And the cosine minimum value is -1. Thus, the cosine of the angle between two vectors determines whether the two sentences have the same meaning. This result is independent of the length of the vectors and is only related to the direction in which the vectors are pointing.

Figure 17 Angle between two vectors A and B



Given two attribute vectors, A and B, the remaining cosine similarity θ is given by the dot product.

$$cos\theta = \frac{AB}{\|A\|\|B\|} = \frac{\Sigma_{i=1}^{n} A_i B_i}{\sqrt{\Sigma_{i=1}^{n} A_i^2} \sqrt{\Sigma_{i=1}^{n} B_i^2}}$$

Here A_i , $B_i$represent each component of vectors A and B respectively. The similarities given range from -1 to 1. And -1 means that the two vectors point in exactly opposite directions, 1 means that they point in the same direction, 0 usually means that they are independent of each other.

CHAPTER IV: EXPERIMENT

Dataset

The dataset we used in our experiments was the Short Answer Scoring V2.0 dataset that was originally collected by Mohler et al (2011). This dataset was primarily extracted from the assignments/exams of the Basic Data Structures of Computer Science at the University of North Texas. The data consists of 81 questions from 10 assignments and two exams. The answers in the dataset are rated on a scale of 0 (completely wrong) to 5 (completely right). The questions, the reference answers, the students' answer to the questions, and their respective grades form the main components of the dataset.

Table 3 Example of Short Answer Scoring V2.0 dataset

Reference Answer: The name of the function and the types of the parameters.				
Index	Student Answer	Score1	Score2	Average
1	It includes the name of the program, the type of parameters it can take. It should also include a brief description of what the function does.	4	5	4.5
2	it includes the specific information about the function such as input and output variable types and how many of each.	1	5	3
3	The function signature includes the name of the function and the types of its arguments.	5	5	5

(Table Continues)

Table 3 Continued

4	It includes the name of the function and the types of its arguments.	5	5	5
5	Name, parameters, scope, and other general function information	4	5	4.5
6	It includes a function name and parameter list. Does not include return type. Function signatures must be different.	5	5	5
7	input parameters and return type	1	5	3
8	The portion of the function prototype that has the function name and the arguments but NOT the return type.	5	5	5
9	Name of the function and the types of its arguments	5	5	5
10	The name of the function and the types of its arguments.	5	5	5
11	Includes the name of the function and the types of its arguments.	5	5	5
12	identification of a function and the data types of its parameters, it has the name, and the data type variables	5	5	5
13	a return type, and input parameters	1	5	3
14	The data type we should return to our main program	0	0	0

There are 2440 student's answers in the dataset. Each answer is a tuple that includes Reference answers, student answers, score1, score2 and average score. Score1 and score2 are manually rated by two different real instructors, and the average score is the average of score1 and score2. We choose 20% of the dataset as the validation set, 20% as the test set, and 60% as the training set.

Because we also want to explore the effect of the text length on the model performance, we divide the dataset into shorter answer set and longer answer set based on the text length of students' answers. Based on the average length of English words is 5 letters (Bochkarev et al, 2012) and considering about the spaces and commas, we divide answers with less than 35 letters as shorter answer set, and those with more than 35 characters are longer answer set.

Table 4 Example of shorter answer set and longer answer set

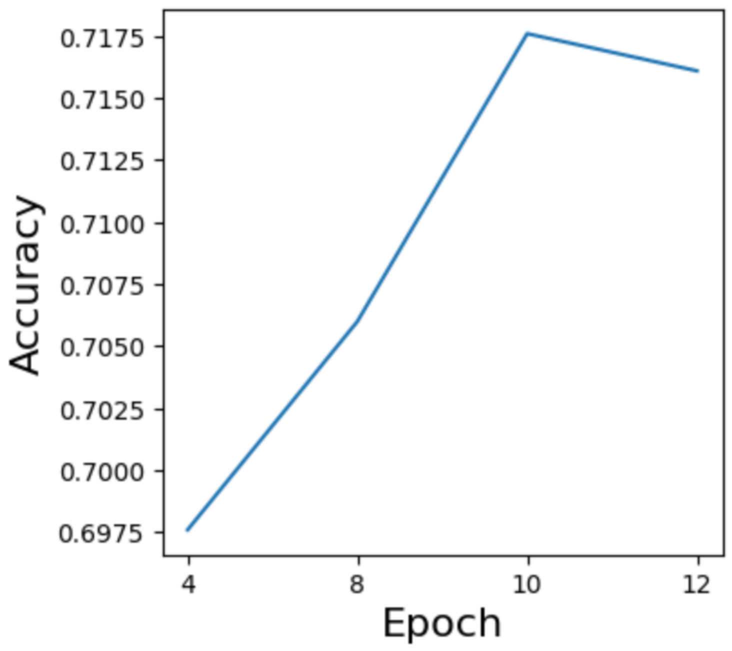
Type	Students' Answer
shorter answer ser	The height of the tree.
longer answer set	A data structure that stores elements following the first in first out principle. The main operations in a queue are enqueue and dequeue.

Experimental Setup

We use the BERT model and Sentence BERT's all-mpnet-base-v2 pre-training model. The all-mpnet-base-v2 model uses a self-supervised contrastive learning objective to train sentence embedding models on a 1B sentence-level dataset and then fine-tune the model to use for tasks such as semantic similarity. It can map sentences and paragraphs to a 768-dimensional dense vector space. It was able to achieve a leading score of 69.57 in Performance Sentence Embeddings.

For fine-tuning, we use the AdamW (Loshchilov et al, 2017) optimizer and set the learning rate to 2e-5. We also set the batch size to 16. To observe the best value of epochs for fine-tuning, we set the epoch to 4, 8, 10, and 12 respectively to determine the best value of epoch.

Figure 18 The result of different Epochs



For the regression task function, the output of the cosine function's ranges is [-1,1]. Since the cosine function determines the similarity based on the cosine of the angle between two input's vectors. So, we can summarize that the similarity of the text is already extremely low when the angle is greater than 90 degrees. Therefore, we set the score of the interval [-1,0.1] to score 0. The rest of the interval is divided equally by the rest 5 labels.

Evaluation Measures

After finishing the modeling, we need to evaluate the effect of the model. We evaluate our ASAG system using 3 measures.

1. Accuracy rate (ACC). Percentage of correctly scored answers.
2. Macro Average F1 Score (M-F1). Calculate metrics for each label and find their unweighted mean. This does not take label imbalance into account.
3. Weighted average F1 score (W-F1): Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance.

Before the introduce the 3 measures, we should know some basic information about measures.

- True Positive: Predicts positive classes as positive classes
- True Negative: predicts negative class as negative class
- False Positive: Predicts the negative class as positive class
- False Negative: predicts positive class as negative class

Table 5 Prediction type

	Predicted Positive (PP)	Predicted Negative (PN)
True	True Positive (TP)	True Negative (TN)
False	False Positive (FP)	False Negative (FN)

The Acc is the percentage of correct predicted results to the total sample:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

Although the accuracy rate can determine the total correct rate, it is not a good indicator of the results when the dataset is unbalanced. As an example, we have a dataset with 90% positive samples and 10% negative samples. And we only need to predict the entire sample as positive to get a high accuracy of 90%. This illustrates that the problem of sample imbalance makes the high accuracy results obtained useless. So we also use the F1 score.

The F1 score is calculated as:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{precision \cdot recall}{precision + recall} \cdot 2$$

$Recall$ reflects the model's ability to identify positive samples, the higher the $recall$, the stronger the model's ability to identify positive samples, and $precision$ reflects the model's ability to discriminate negative samples, the higher the $precision$, the stronger the model's ability to discriminate negative samples. F1 score is a combination of the two. the higher the F1 score, the more robust the classification model is.

F1 score only works for the binary classification task, while Macro Average F1 Score and Weighted average F1 score are relative to multi-label categories. For Micro-F1, calculate metrics for each label and find their unweighted mean. Weighted average F1 score calculate metrics for each label and find their average weighted.

CHAPTER V: RESULT AND ANALYSIS

It is commonly believed that datasets with multiple integer labels are more suitable for multiclassification tasks. In our test dataset, the cosine function can predict the real label of 376 cases. The softmax function can predict the real label of 356 cases, in which 317 cases are predicted correctly by both functions. In addition to that, the cosine function can predict more 20 cases correctly than softmax function. Therefore, we can make a conclusion that with a suitable task function, a multiclassification task can be converted to a regression task with better results in ASAG domain.

Table 6 Result of different task functions

	ACC	M-F1	W-F1
softmax	0.73	0.54	0.46
cosine	0.77	0.56	0.48

We can see that Sentence BERT obtained impressive results which reached an accuracy of more than 70 percent on the dataset, indicating that the pre-trained model has an excellent adaptation to short answer scoring area.

Table 7 Result of SBERT and BERT

	ACC	M-F1	W-F1
vs Score1	0.73	0.54	0.49
vs Score2	0.74	0.56	0.47
vs Ave	0.78	0.57	0.51
BERT	0.73	0.61	0.54

Also, we compare the prediction results of the model with three different labels (score1, score2 and ave), while the variance rate is within 5%. This further demonstrates the reliability of this pre-trained model. We also compare with the BERT model, and we can see that the sentence BERT still has a 5% accuracy improvement, which shows that in the field of sentence similarity, the pre-trained model after training on a task-relevant dataset is better than the generic BERT model.

Finally, the accuracy, MF1, and WF1 of the shorter answer set are much higher than those of the longer answer set.

Table 8 Result of shorter answer set and longer answer set

	ACC	MF1	WF1
shorter answer set	0.81	0.71	0.63
longer answer set	0.67	0.49	0.44

we conclude that shorter answer set can get better results than those with more letters on our model. This result is not surprising considering the nature of word embedding. As the excessive information to be stored in the longer answer sets' vector, the vector discards some information, such as the location information of certain words, but the location information may be very important in the model operation. As a result, the discarded information causes a decrease in accuracy.

CHAPTER VI: DISCUSSION

This work shows that a pre-trained neural network model with a proper task function can achieve promising results in automatic short answer grading using only a small amount of data. Specifically, the Sentence BERT model pre-trained on the STS dataset has a better performance in ASAG than the generic BERT model. Since our model achieves a better result on the shorter answer dataset, we believe limiting the length of answers plays an important role on the model's performance. However, due to the black box nature of deep learning, we are unable to explain the reason for scoring. It means that the model lacks the ability to interpret the process of identifying errors in answers. Additionally, we only tested the short answer question dataset in computer science domain. In future research, we expect to validate our model by using datasets from other domains. It is also possible to further explore how to improve the performance of the model on longer textual short-answer question datasets.

REFERENCES

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.

Bentivogli, L., Clark, P., Dagan, I., & Giampiccolo, D. (2009, November). The Fifth PASCAL Recognizing Textual Entailment Challenge. In TAC.

Burrows, S., Gurevych, I., & Stein, B. (2015). The eras and trends of automatic short answer grading. International Journal of Artificial Intelligence in Education, 25(1), 60-117.

Bailey, S., & Meurers, D. (2008, June). Diagnosing meaning errors in short answers to reading comprehension questions. In Proceedings of the third workshop on innovative use of NLP for building educational applications (pp. 107-115).

Bochkarev, V., Shevlyakova, A. V., & Solovyev, V. D. (2012). Average Word Length Dynamics as Indicator of Cultural Changes in Society, August.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., & Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. arXiv preprint arXiv:1708.00055.

Chen, Z., Zhang, H., Zhang, X., & Zhao, L. (2018). Quora question pairs. URL https://www. kaggle. com/c/quora-question-pairs.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Dolan, W. B., & Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In Proceedings of the Third International Workshop on Paraphrasing (IWP2005).

Gütl, C. (2007, April). e-Examiner: towards a fully-automatic knowledge assessment tool applicable in adaptive e-learning systems. In Proceedings of the Second International Conference on Interactive Mobile and Computer Aided Learning (pp. 1-10).

Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In Text summarization branches out (pp. 74-81).

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Mohler, M., Bunescu, R., & Mihalcea, R. (2011, June). 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: Human language technologies (pp. 752-762).

Madnani, N., Burstein, J., Sabatini, J., & O'Reilly, T. (2013). Automated Scoring of Summary-Writing Tasks Designed to Measure Reading Comprehension. Grantee Submission.

Page, E. B. (1966). The imminence of... grading essays by computer. The Phi Delta Kappan, 47(5), 238-243.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics (pp. 311-318).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

Pulman, S., & Sukkarieh, J. (2005, June). Automatic short answer marking. In Proceedings of the second workshop on Building Educational Applications Using NLP (pp. 9-16).

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science.

Ramshaw, L. A., & Marcus, M. P. (1999). Text chunking using transformation-based learning. In Natural language processing using very large corpora (pp. 157-176). Springer, Dordrecht.

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250.

Sung, C., Dhamecha, T. I., & Mukhi, N. (2019, June). Improving short answer grading using transformer-based pre-training. In International Conference on Artificial Intelligence in Education (pp. 469-481). Springer, Cham.

Sung, C., Dhamecha, T., Saha, S., Ma, T., Reddy, V., & Arora, R. (2019, November). Pre-training BERT on domain resources for short answer grading. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 6071-6075).

Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. arXiv preprint cs/0306050.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013, October). Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing (pp. 1631-1642).

Tulu, C. N., Ozkaya, O., & Orhan, U. (2021). Automatic Short Answer Grading With SemSpace Sense Vectors and MaLSTM. IEEE Access, 9, 19270-19280.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Williams, A., Nangia, N., & Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. arXiv preprint arXiv:1704.05426.

Warstadt, A., Singh, A., & Bowman, S. R. (2019). Neural network acceptability judgments. Transactions of the Association for Computational Linguistics, 7, 625-641.

Wang, W., Yan, M., & Wu, C. (2018). Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. arXiv preprint arXiv:1811.11934.

Wolfram, S. (2017). Wolfram language & system documentation center. Retrieved from https://reference.wolfram.com/language/ref/Tanh.html

Zellers, R., Bisk, Y., Schwartz, R., & Choi, Y. (2018). Swag: A large-scale adversarial dataset for grounded commonsense inference. arXiv preprint arXiv:1808.05326.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In Proceedings of the IEEE international conference on computer vision (pp. 19-27).

APPENDIX: PARTIAL EXPERIMENTAL RESULTS

Cosine:
Reference answer: The Euler tour traversal of a tree is a specific way of navigating a tree that involves following the tree starting at the very top and moving along the left side of the tree first, cupping in to visit the parents of children nodes. It allows for each node to be visited from the left, the right and the bottom.<br><br>The Euler tour first progresses to a left child if there is one, then progresses to it's parent, then it's next child, then it's parent's parent.
Student answer: A walk around the tree, starting with the root, where each node is seen three times: from the left, from below, from the right.
value of cosine: tensor([[0.8518]], device='cuda:0')
score1: 5

Reference answer: the ampersand (&) means "pass by reference". When the function is called, a pointer to the variable, instead of the variable itself, will be passed into the function.
Student answer: The memory address of its operand.
value of cosine: tensor([[0.9035]], device='cuda:0')
score1: 3

Reference answer: the name of the function and the types of its arguments
Student answer: The name of the function and the types of the parameters.
value of cosine: tensor([[0.9744]], device='cuda:0')
score1: 5

Reference answer: Private, Public, Protected, or Friend.
Student answer: Private and public.
value of cosine: tensor([[0.9034]], device='cuda:0')
score1: 5

Reference answer: A header file usually contains class and/or function prototypes.
Student answer: To store a class interface, including data members and member function prototypes.
value of cosine: tensor([[0.8358]], device='cuda:0')
score1: 4

Reference answer: A constructor initialized values at the execution of its instantiation. It provides default values.
Student answer: A constructor is called whenever an object is created, whereas a function needs to be called explicitly. Constructors do not have return type, but functions have to indicate a return type.
value of cosine: tensor([[0.7658]], device='cuda:0')
score1: 4

Reference answer: in postfix format
Student answer: First, they are converted into postfix form, followed by an evaluation of the postfix expression.

```
value of cosine: tensor([[0.7575]], device='cuda:0')
score1: 3

Reference answer: Using an index outside the bounds of the array generates a run-time error.
Student answer: Run-time error.
value of cosine: tensor([[0.7569]], device='cuda:0')
score1: 5

Reference answer: Cycle through the unsorted list, place the minimum in the next slot in the
sorted list, and repeat.
Student answer: Taking one array element at a time, from left to right, it identifies the minimum
from the remaining elements and swaps it with the current element.
value of cosine: tensor([[0.9371]], device='cuda:0')
score1: 5

Reference answer: local, global, local-variable, function specific
Student answer:  Private and public.
value of cosine: tensor([[0.6278]], device='cuda:0')
score1: 3

Reference answer: Starting at the root node, it branches off into one or two subsets that are
binary subtrees of the root.<br><br>Each node has at most two children, the left child and the
right child.
Student answer: A tree for which the maximum number of children per node is two.
value of cosine: tensor([[0.8478]], device='cuda:0')
score1: 5

Reference answer: A single Element on the Array.
Student answer: When the size of the array to be sorted is 1 (or 2)
value of cosine: tensor([[0.9514]], device='cuda:0')
score1: 5

Reference answer: They differentiated by the compiler by the conditions/inputs used for one of
the overloaded functions.
Student answer: Based on the function signature. When an overloaded function is called, the
compiler will find the function whose signature is closest to the given function call.
value of cosine: tensor([[0.9093]], device='cuda:0')
score1: 4

Reference answer: compilation error
Student answer: Run-time error.
value of cosine: tensor([[0.0964]], device='cuda:0')
score1: 0
```