Engineering Applications of Artificial Intelligence 123 (2023) 106292

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Improving the performance of automatic short answer grading using transfer learning and augmentation

Sridevi Bonthu [a,*], S. Rama Sree [b], M.H.M. Krishna Prasad [a]

[a] Computer Science and Engineering Department, Jawaharlal Nehru Technological University, Kakinada, Andhra Pradesh, India
[b] Computer Science and Engineering Department, Aditya Engineering College, Surampalem, Andhra Pradesh, India

ARTICLE INFO

Keywords:
ASAG
Transfer learning
SBERT
Augmentation
SPRAG corpus
Random swap
Random deletion
Synonym replacement
Backtranslation

ABSTRACT

The task of grading answers ranging from one phrase to one paragraph using computational techniques is known as Automated Short Answer Grading (ASAG). The performance of existing systems is not good enough due to limited data and the lack of availability of data in many domains. Many ASAG systems were developed as an outcome of the active research in this field. This study builds an effective system for grading short answers in the programming domain by leveraging Pre-trained Language Models and Text Augmentation. We fine-tuned three-sentence transformer models on the SPRAG corpus with five different augmentation techniques: viz., Random Deletion, Synonym Replacement, Random Swap, Backtranslation, and NLPAug. The SPRAG corpus contains student responses involving keywords and special symbols. We experimented with four different data sizes with the augmented data to determine the impact of training data on the fine-tuned sentence transformer model. this paper provides an exhaustive analysis of fine-tuning pretrained sentence transformer models with varying sizes of data by applying text augmentation techniques. we found that applying random swap and synonym replacement techniques together while fine-tuning has given a significant improvement, With a 4.91% increase in accuracy and a 3.36% increase in the F1-score. All the trained models are publicly available[1].

1. Introduction

Research in the education domain using Artificial Intelligence is moving in the right direction and benefiting all stakeholders (Wilianto and Girsang, 2023). The Automated Short Answer Grading (ASAG) task is a problem in educational research using AI as it reduces the burden on humans to evaluate the descriptive answers. The task of ASAG is not a new problem. It is formally defined as assigning a score to the student response based on one or more reference answers. It is a supervised learning problem and is addressed as either a regression problem or a classification problem (Burrows et al., 2015).

The ASAG task has many developments, ranging from traditional approaches to neural networks (Burrows et al., 2015). The prior work focused purely on textual student responses. While handling the ASAG task, there is a requirement to focus on specific domains such as mathematics, physics, programming, etc., as these domains contain both textual and non-textual characters such as symbols (Zhang et al., 2022). The adoption of recent technologies is important to solve ASAG with respect to this dimension. The goal of this work is to create an efficient system for grading short programming-related answers using Pretrained Language Models (PLM) and text augmentation techniques. We are building a binary classification model on the SPRAG dataset with a learning objective to assign a grade 0 or 1. The dataset D consists of N examples. $D = \{(X_i, y_i)|i = 1, 2, \ldots, N\}, X_i = \{q_i, sa_i, ra_i\}, y_i \in \{0, 1\}$, where q_i, sa_i, and ra_i are raw text representing the question, student answer, and the reference answer respectively. The objective is to find $P(X_i|y_i)$ and assign a grade 0 or 1.

One of the main challenges with ASAG tasks is the limited availability of the data and the non-availability of data in many domains (Bonthu et al., 2021). In practice, the majority of the datasets have limited graded answers, as manual grading is labor intensive (Lun et al., 2020). Building a classification model with limited data is challenging. This issue can be addressed through the domain knowledge acquired by the PLMs. The PLMs are pretrained with large datasets related to many domains, and they need to be fine-tuned with in-domain datasets specific to the task. The amount of data required to fine-tune PLMs is minimal when compared with pretraining from scratch (Grangier and Iter, 2021). Further, augmentation techniques can be applied to training data, as fine-tuning PLMs on ASAG data alone would not be able to generalize well. This paper focuses on the ASAG task in the programming domain. The responses in this domain involve different ways of presenting and reasoning. The key challenge of this work is that both the reference and student answers are combinations of

* Corresponding author.
E-mail addresses: sridevi.db@gmail.com (S. Bonthu), ramasrees@aec.edu.in (S. Rama Sree), krishnaprasad.mhm@gmail.com (M.H.M. Krishna Prasad).

[1] https://github.com/sridevibonthu/SPRAG/tree/main/augmentation

https://doi.org/10.1016/j.engappai.2023.106292
Received 20 September 2022; Received in revised form 9 March 2023; Accepted 4 April 2023
Available online 12 April 2023
0952-1976/

Table 1
Dataset examples showing the question, reference answer, and two student answers along with the obtained grade from the Python programming course. A normalized score between 0 and 1 is reported for regression tasks and a binary label 0, 1 for classification tasks.

Example - 1	
Question	How to create anonymous functions in python?
Reference answer	Anonymous functions are created using lambda keyword in python.
Student response-1	Anonymous function is nothing but the lambda functions. It creates by using keyword lambda. It doesn't have name. It can take multiple arguments. It doesn't have return function. (1)
Student response-2	By using def (0)
Example - 2	
Question	Write about the division operators of Python.
Reference answer	We have two division operators in python. They are "/" and "//". "//" is called as floor division and it gives the floor value of the obtained result. Whereas "/" operator gives the exact result. For example, 5//2 gives 2 and 5/2 gives 2.5
Student response-1	There are 2 division operators. (1) floor division (//)-it returns the quotient of the division leaving the remainder behind. (2) normal division (/)-it returns the quotient of the division when the remainder becomes 0. (1)
Student response-2	Division operator can divide the numbers. (0)

natural language, keywords, identifiers, special symbols, expressions, etc. Table 1 shows two sample examples of reference answers and student responses in their raw textual form. The existing PLMs are not trained in technical or programming jargon. Fine-tuning the PLM with limited domain-specific data results in overfitting. This study provides an exhaustive analysis of the usage of text augmentation in fine-tuning the PLM with limited domain-specific data.

Many researchers confirmed that the performance of the ASAG systems is dependent on the amount of training data (Wilianto and Girsang, 2023). To test the role of the size of training data while fine-tuning PLM with augmented data, we trained every model with four different sizes of data (low to high). This paper proposes an ASAG framework for grading programming related questions. The framework mainly involves fine-tuning a few PLMs with five different augmentation strategies, viz., Random Deletion, Synonym Replacement, Random Swap, BackTranslation, and NLPAug. The rest of the paper refers to these techniques as AugFs. The main contributions to this work are:

- Fine-tuning SBERT, a sentence transformer, to generate rich embedding with three different PLMs, to get a better representation of the sentences.
- To further enhance the sentence embeddings, five augmentation strategies, AugFs are investigated.
- Compares the performance of the original and the augmented data on three PLMs.
- Performs ablation studies to suggest a practical solution for ASAG.

The rest of the paper is structured in the following way. Section 2 provides an overview of the academic work carried out in the ASAG and augmentation domains. Section 3 presents the details of the adopted augmentation strategies. Section 4 presents the method, while Section 5 provides the experimentation details. In Section 6, the results are evaluated and an ablation study is performed.

2. Related work

This section starts by introducing the work that happened in the area of ASAG, then quotes about how transfer learning changed the landscape of NLP, and closes with the prior work on augmenting text data.

Natural Language Processing (NLP) is a subfield of AI and gaining lot of focus on research due to the emergence of an enormous number of applications (Chapelle and Chung, 2010). The research in ASAG has taken a new shape due to the advancements in NLP. The ASAG task is addressed by researchers using traditional, machine learning (ML) and deep learning (DL) approaches (Bonthu et al., 2021). The classical ML approaches used handcrafted features like features based on criteria, lexical similarity features, n-gram based features, graph alignment features, cardinality based text overlapping features, vectorized features using Bag-of-Words, TF–IDF, shallow lexical features, and vector-based similarity features (Mohler et al., 2011; Heilman and Madnani, 2013; Jimenez et al., 2013; Dzikovska et al., 2013). Recently, DL-based approaches are outperforming traditional and ML-based approaches. The various DL approaches employed by researchers in the past few years are Long Short Term Memory (LSTM) networks, Bi-directional LSTMs, Transformers, Siamese Networks, Transfer Learning, Autoencoders, and Attention. A Siamese Bi-LSTM based on Earth Mover Distance (EMD) for the regression task was proposed by Kumar et al. and it has gained a significant improvement over the classical approaches (Kumar et al., 2017). Efficient memory networks are proposed by Zhao et al. (2017) to improve automated scoring. Riordan et al. (2017) used multiple architectures based on Bi-LSTM on publicly available datasets to explore the effectiveness of attention. A Multi-Domain neural model architecture is proposed using LSTM and attention to generalize on domain-specific aspects (Conneau et al., 2017). Multi-way attention and Manhattan LSTM (Tan et al., 2018) have become successful in improving the performance. BERT-based models are effectively used by Sung et al. to grade the answers in low-resource settings. They have fine-tuned transformer-based models and showed a significant improvement in the results (Sung et al., 2019).

Transfer Learning extends the traditional approach of learning in isolation by leveraging data from additional domains. This technique results in better generalization of the model. The field of NLP has witnessed the success of several transfer learning approaches. It significantly improved the state-of-the-art on several NLP tasks (Ruder et al., 2019), ASAG is not an exception. Consequently, sentence encoders started outperforming the BoW and TF–IDF based models. To cut down on the massive computational cost involved in sentence encoding, Reimers and Gurevych (2019) proposed the SBERT architecture. The adjustments made in SBERT enable it to be used for clustering, semantic search, and semantic similarity. SBERT is successful with the ASAG task too (Condor et al., 2021).

Augmentation increases the size of training data and thereby improves the performance of the model. Like vision, text augmentation has no standard approaches and less common in NLP. Text data augmentation has been widely used in various applications in recent years to improve the performance of NLP tasks such as text classification, natural language generation, named entity recognition, question answering, paraphrasing, and semantic parsing by increasing the size and diversity of the training data (Bayer et al., 2022). Despite substantial research in recent years, text augmentation still faces many difficulties (Bayer et al., 2022). Prior work started by switching out words and substituting them with synonyms from the WordNet database (Zhang et al., 2015). Wang et al. replaced the word embedding vectors with similar vectors based on cosine distance (Wang and Yang, 2015). Grammar induction (Jia and Liang, 2016) and task-specific rules (Silfverberg et al., 2017) are added to training examples to boost the model's generalizability. Significant improvement is observed by generating

Table 2
Application of the adopted augmentation techniques (AugFs) on two student responses from the SPRAG dataset.

Method	Resultant text
Example-1	
Original	A compiler is a computer program it is used to convert programming language code to machine level language code.
RD	Compiler is a computer program it is used to convert programming language code machine level language code.
SR	A compiler is a computer program it is employ to change programming language inscribe to machine level language code.
RS	A compiler is a computer program it is language to convert programming language to code machine level used code.
BT	The compiler is a computer program that converts the programming language code into a machine-level language code.
NA	A cK, piker is a computer program it is seS to convert programming language Xpde to mafT ∗ ne level <anhuags dKde.
Example-2	
Original	Dictionary is an unordered collection of key–value pairs. Keys are immutable objects and values can be mutable or immutable. An empty dictionary is represented with {} or dict() in python.
RD	Dictionary is an unordered collection of pairs. Keys are immutable objects and values can be mutable or immutable. An dictionary is represented with {} or dict() in python.
SR	Dictionary is an unordered collection of key–value pairs. keystone are immutable objects and values can be mutable or immutable. An evacuate dictionary is represented with {} or dict() in python.
RS	Dictionary unordered an is collection of key–value pairs. unique are immutable objects and values can be mutable or immutable. An empty dictionary is represented with {} or dict() in python.
BT	Dictionary is an disorderly collection of key values. The key is an unchanged object, and the value may be variable or variable. Python uses {} or dict() to represent an empty dictionary.
NA	Dictionary is an unordered collection 0f key–value pairs. Keys are immutable objects and values can be mutable or immutable. An empty dictionary is represented with {} 0k dict() in python.

new data by running two machine translators back to back (Xia et al., 2017). There is no significant augmentation experimentation carried out on multilingual and code-mixed data.

In addition to the previous works, our work investigates and analyzes the effectiveness of augmentation techniques in fine-tuning sentence transformer models on an ASAG task with programming-related data.

3. Data augmentation strategies

Data Augmentation is a technique to enlarge the available training data by applying various label-preserving transformations (Taylor and Nitschke, 2018). It transforms an example x_i to $\hat{x}_i$, which retains most of the properties of x_i. There is no standard set of augmentation techniques that achieve state-of-the-art performance in NLP applications. Many researchers are experimenting with label-preserving augmentation techniques in sentiment analysis (Bonthu et al., 2022; Wei and Zou, 2019), machine translation, etc. We have picked a few of them to improve the performance of the ASAG task. This section provides a brief overview of the five augmentation strategies used in our experimentation. They are Random Deletion(RD), Synonym Replacement(SR), Random Swap(RS), BackTranslation (BT), and NLPAug Library(NA). The Table 2 presents the augmented examples using the adopted techniques for two training examples.

3.1. Random deletion

Random Deletion (RD) creates a new augmented example $\hat{x}$ by randomly deleting n number of words from the supplied training example x with a probability p (Wei and Zou, 2019). If the training example $x = (x_1, x_2, ..x_m)$ has m tokens then the resultant will contain $m - n$ tokens, $(x_1, x_2, ..x_{m-n})$. This approach may produce meaningless training examples if applied to short sentences or if the value of p is large.

3.2. Synonym replacement

The Synonym Replacement (SR) approach is a word replacement technique that replaces the tokens of the sentence with their synonyms (Wei and Zou, 2019). The synonyms will be retrieved from a valid thesaurus or from a lexical database like WordNet (Miller, 1995). The SR technique is successfully utilized for more applications like machine translation (Gao et al., 2019), named entity recognition (Dai and Adel, 2020), etc.

3.3. Random swap

The Random Swap (RS) selects two tokens randomly from the input example x and swaps them to generate the augmented example $\hat{x}$ (Wei and Zou, 2019). The swapping can be repeated for n number of times. The value of n should be minimal to avoid the loss of meaning of the training examples. The augmented examples with nouns generated through this approach may fool the model with adversarial text attacks. For example, transformation of *"Rama killed Ravana"* into *"Ravana killed Rama"* is a text attack.

3.4. BackTranslation

BackTranslation (BT) is a simple and easy-to-use augmentation technique to obtain additional training data (Xia et al., 2017; Sennrich et al., 2015). It uses existing machine translation algorithms in two steps: (1) Forward Translation (from source to target); and (2) Backward Translation (from target to source). This dual translation is expressed in Eq. (1), where L_S is the source language and L_T is the target language.

$$\hat{x} = translate(translate(x, L_S \rightarrow L_T), L_T \rightarrow L_S) \quad (1)$$

BT is computationally expensive due to this dual translation. Two simple python libraries available to backtranslate text between two languages are GoogleTrans.[2] and Baidu Translation API[3]

3.5. NLPAug

NLPaug[4] is a lightweight python library to generate synthetic data. We have used KeyboardAug augmenter to randomly select and replace characters present on the keyboard. This technique mainly injects noise into the training example.

4. Method

The main objective of this work is to test the effectiveness of augmentation on fine-tuning sentence transformers for ASAG tasks in both low-resource to complete data settings. To accomplish that, three different sentence transformers are fine-tuned by varying the training sizes. All the three models are trained four times with 25%, 50%, 75%, and 100% of the training data and analyzed for their performance. The entire work flow followed by this work is illustrated in Fig. 1.

[2] https://pypi.org/project/googletrans/.
[3] https://pypi.org/project/baidu-trans/.
[4] https://github.com/makcedward/nlpaug.

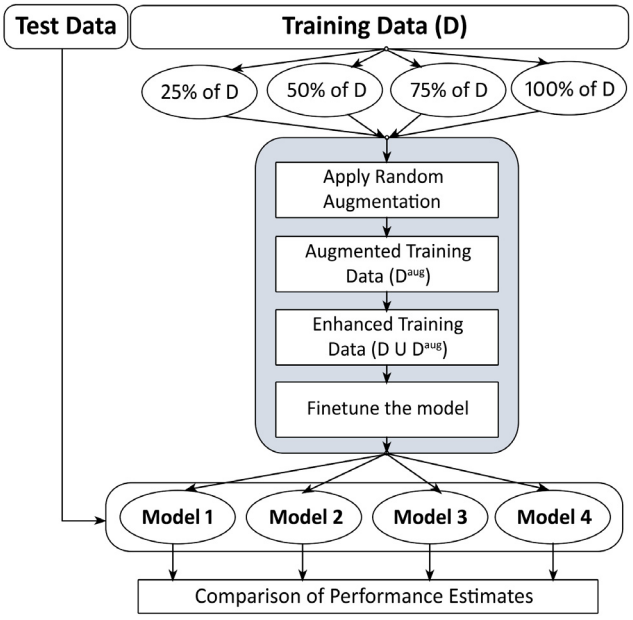


Fig. 1. Framework for fine-tuning a sentence transformer with Pre-trained Language Model. The model is fine-tuned with four different training sizes.

An SBERT architecture (Reimers and Gurevych, 2019) with a pre-trained language model (PLM) is used for training. The SBERT model finetunes a pre-trained transformer (BERT) to yield useful sentence embeddings. SBERT also adds a pooling operation on top of the transformer output and produces fixed size vectors for the input sequences. To produce semantically meaningful embeddings, the SBERT model is fine-tuned using a siamese network (Dong and Shen, 2018) and then the embeddings are compared with cosine-similarity. Let sa and ra be the sentence embeddings generated after the mean pooling layer for the student answer and reference answer, respectively. To perform training with a classification objective, the embeddings *sa, ra*, and their element-wise difference $|sa - ra|$ are concatenated and multiplied with a trainable weight vector (W). The dimensions of the vector W are $3nX2$, where n refers to the dimension of the embedding and 2 to the number of labels. The outcome of the classification is obtained by a softmax layer. The entire calculation is expressed in Eq. (2).

$$o = softmax(W(sa, ra, |sa - ra|)) \tag{2}$$

To perform regression, cosine similarity between the two embeddings sa and ra, which results in a range of -1 to 1, is employed. The cosine similarity between the two vectors sa and ra is computed using Eq. (3) where $sa \in \mathbb{R}^n$ and $ra \in \mathbb{R}^n$.

$$\begin{aligned} cosine\_similarity(sa, ra) &= \frac{ra \cdot sa}{\|ra\| \times \|sa\|} \\ &= \frac{\sum_{i=1}^{n} ra_i sa_i}{\sqrt{\sum_{i=1}^{n} ra_i^2}\sqrt{\sum_{i=1}^{n} sa_i^2}} \end{aligned} \tag{3}$$

The experimentation has used *stsb-distilbert-base*,[5] *paraphrase-albert-small-v2*,[6] and *quora-distilbert-base*[7]. pre-trained sentence transformer models. *stsb-distilbert-base* and *quora-distilbert-base* models are sentence-transformers used for semantic search or clustering by mapping the sentences to a dense 768-dimensional vector space. They follow the DistilBERT architecture, which is a faster, cheaper, and smaller version of BERT to pre-train (Sanh et al., 2019). *stsb-distilbert-base* is trained on the STS, SNLI, and MultiNLI benchmark datasets, whereas *quora-distilbert-base* is trained on Quora Question bank dataset. *paraphrase-albert-small-v2* is also a sentence transformer trained using SNLI, MultiNLI, and wiki-atomic-edits datasets. Its base model is the ALBERT (Lan et al., 2019), which is a shrunken BERT model without compromising the performance.

Data:
Dataset D
Set of Augmentation functions *AugFs*

Function AugData(D, $AugFs$):
  for $(x_i, y_i) \in D$ **do**
    $r_f \leftarrow random(AugFs)$;
    $\hat{x_i} \leftarrow [r_f(SA_i), RA_i]$; /* $(SA_i, RA_i) \in x_i$ */
    $D_i^{aug} \leftarrow (\hat{x_i}, y_i)$;
  end
  $D' \leftarrow D \cup D^{aug}$;

  return D';

Algorithm 1: An algorithm for augmented data generation

The Algorithm 1 titled $AugData$ is used to increase the training examples by applying various augmentation techniques. It takes the training data D and a list of $AugFs = [RD, SR, RS, BT, NA]$ as inputs. The augmentation techniques adopted are presented in Section 3. Let the original data D be $(x_i, y_i)_{i=1}^n$ where $x_i = (RA_i, SA_i)$. The Augmentation technique specified in Algorithm 1 is applied on student response to generate augmentation data D^{aug}. The augmented data $D^{aug} = (\hat{x_i}, y_i)_{i=1}^n$ where $\hat{x_i} = (RA_i, \hat{SA_i})$. Few student responses after applying the $AugFs$ are tabulated in the Table 2.

4.1. Evaluation metrics

The results of the proposed method are reported in both *accuracy* and *F1-score*. The metrics to evaluate a classification model are obtained from the confusion matrix. The main elements of the confusion matrix with respect to the *ASAG* task, where the student-authored response (SA) can be either *correct (positive)* or *incorrect (negative)*, are defined as follows:

- True Positives (**TP**) are the number of student responses that are actually correct and classified as correct.
- False Positives (**FP**) are the number of student authored responses that are actually incorrect but classified as correct.
- False Negative (**FN**) are the number of student authored responses that are actually correct but classified as incorrect.
- True Negatives (**TN**) are the number of student responses that are actually incorrect and classified as incorrect.

Although accuracy is the most commonly used general metric for classification, it cannot deal with class distribution imbalance. It can be computed using the formula in Eq. (4).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4}$$

The objective of a good classifier is to obtain high precision and recall values (i.e., no or few FP or FN). The F1-score expresses these metrics in a single metric, as there is a trade-off between precision and recall. The F1-score is computed using Eq. (5).

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5}$$

where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$.

[5] https://huggingface.co/sentence-transformers/stsb-distilbert-base.
[6] https://huggingface.co/sentence-transformers/paraphrase-albert-small-v2.
[7] https://huggingface.co/sentence-transformers/quora-distilbert-base.

5. Experiments

Several comparative experiments were conducted to evaluate the effectiveness of augmentation strategies in ASAG tasks on varying training data sizes (from low to high). This section introduces the dataset and evaluation metrics used, experimentation settings, and the implementation details. The dataset and the source code are publicly available.[8]

5.1. Dataset

The ASAG task is evaluated on several datasets which involve answers written in English or some other language. This work used a Short Programming Related Answer Grading Dataset (SPRAG). This dataset contains reference and student answers involving Python programming concepts and libraries. The major challenge with this dataset is finding the semantic similarity between the reference and student answers, as they involve symbols, digits, keywords, variables, etc. The dataset size is 4039 records (2707 correct and 1332 incorrect) and they are responses to 114 different questions. 85% of the data is used for training and rest for testing the models. The questions are distributed into five different categories, viz., apply, difference, remember, define, and explain.

It is a binary classification problem and dependent on similarity between the RA and SA. The results are reported in two ways. Accuracy and F1-score are measured for classification; Pearson and Spearman correlation coefficients are measured for similarity.

5.2. Experiment settings

The Transformer models with attention mechanism rebuilt the landscape of NLP. The sentence-transformers are able to produce rich and dense sentence embeddings, which can be further used to compare similarities between the sentences. These rich embeddings can be applied to a variety of scenarios, including semantic search, Semantic Textual Similarity (STS), and clustering. Three pre-trained sentence transformer models are adopted for experimentation. These models are fine-tuned using the SPRAG dataset. Every model is fine-tuned and tested with four different training data sizes (25%, 50%, 75%, and 100%) to ensure that it performs well or not in low-resource settings as well. The seven different combinations of the training data fed to the models for checking the effectiveness of augmentation are listed below. All the models are trained with varying training sizes and augmentations.

1. SPRAG dataset
2. SPRAG + AugFs
3. SPRAG + RD
4. SPRAG + SR
5. SPRAG + RS
6. SPRAG + BT
7. SPRAG + NA

For each of the seven combinations listed above, every model is trained with different size of data. The total number of trained instances of the models are $28(7X4)$. A classification model built on top of SBERT is fitted to these input types. With the three models, we have 84 results to compare and understand. Thus, we compare 84 different versions of output with two kinds of evaluation.

The training data is generated by using Algorithm 1 with specific parameters for *AugFs*. The parameters are identified based on the length of the text and the context. The SR method finds the synonyms from the wordnet database through the synset interface and replaces up to three random words with their synonyms if the number of words in the student response is greater than five. The RD method randomly selects words with a probability of 20% ($p = 0.2$) if the student response has more than two words. The RS method randomly selects two pairs ($n = 2$) and swaps them if the number of words in the student response is more than four. In the backtranslation method, the source language (L_S) is English and the target language (L_T) is Chinese. Finally, in the NLPaug method, we used OCRaug[9] to generate noise characters for the examples. If the original data contains D records, then Algorithm 1 results in $2 \times D$ records.

For better comparison, common hyperparameters are used on all 84 input types for model training. Pretrained sentence transformer models from the Huggingface library are chosen to test the effectiveness of augmentation. The models are trained for 10 epochs with a batch size of 16. Training uses the WarmupLinear scheduler and AdamW (Loshchilov and Hutter, 2018) optimizer. 10% of the training data is used for warming up. The maximum learning rate is 2e−05. A binary classification evaluator is used to save the best model while training. To assess the performance on test data, both classification and similarity evaluators are utilized. The first evaluation reports accuracy and F1 score. The second evaluation reports the Spearman rank correlation between the cosine-similarity of the embeddings of (RA, SA) and the gold labels. All the experiments are fine-tuned using the NVIDIA Qudro GV100 GPU with 32 GB of RAM.

Table 3
Results on the SPRAG dataset. The accuracy (Acc.) and F1-Score (F1) are reported in percentage with varying training data sizes on the SBERT with three pre-trained language models.

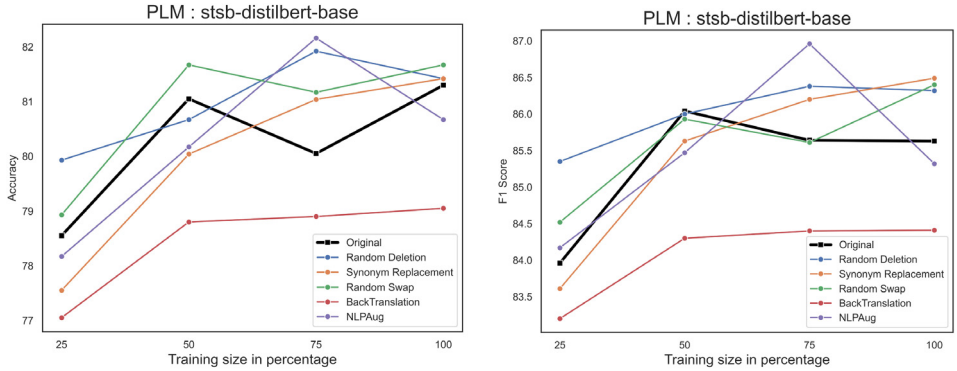
Data	25%		50%		75%		100%	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
stsb-distilbert-base								
Original	78.55	83.96	81.05	86.04	80.05	85.64	81.3	85.63
AAug	78.3	84.09	79.42	85.09	79.92	85.03	79.92	85.07
RD	**79.93**	85.35	80.67	86	81.92	86.38	81.42	86.32
SR	77.55	83.61	80.04	85.63	81.04	86.2	81.42	86.49
RS	78.93	84.52	**81.67**	85.93	81.17	85.61	**81.67**	86.4
BT	77.05	83.2	78.8	84.3	78.9	84.4	79.05	84.41
NA	78.17	84.17	80.17	85.47	**82.16**	86.96	80.67	85.32
paraphrase-albert-small-v2								
Original	78.55	83.8	78.93	84.65	79.93	84.44	79.3	84.75
AAug	78.8	84.3	78.67	84.4	79.05	84.18	79.42	84.76
RD	78.43	83.78	**79.8**	85.02	79.8	85.2	80.55	85.39
SR	**80.79**	85.76	79.55	84.94	80.4	85.3	80.17	85.42
RS	78.93	84.22	**79.8**	85.09	**80.92**	85.01	80.42	85.34
BT	76.05	81.9	78.5	84.1	78	83	78	83
NA	78.92	84.01	78.67	84.3	79.55	84.5	**80.92**	86
quora-distilbert-base								
Original	80.55	85.3	**81.42**	85.74	81.92	86.66	82.54	86.63
AAug	80.17	84.87	80.17	85.32	79.05	84.36	81.29	85.9
RD	**80.67**	85.66	80.42	85.24	**82.13**	86.49	**83.1**	86.75
SR	80.54	85.58	80.29	85.13	81.9	85.7	80.54	85.09
RS	79.43	85.39	80.8	85.79	80.55	85.34	81.05	85.24
BT	78.17	83.4	80.1	84.5	79	84	79.9	84.8
NA	80.54	85.79	80.67	86.22	80.79	85.84	81.17	85.6

6. Results and discussions

The SBERT architecture with the three considered PLMs fine-tuned on the SPRAG dataset is considered the baseline result. This section analyzes the results and compares the effectiveness of the augmentation in the improvement of the results. First, the results of the AugFs with individual augmentations were examined. Then, an ablation study on the combination of best-performing techniques is analyzed to find an effective augmentation strategy. Finally, the SBERT fine-tuned with augmented data is compared with the baseline.

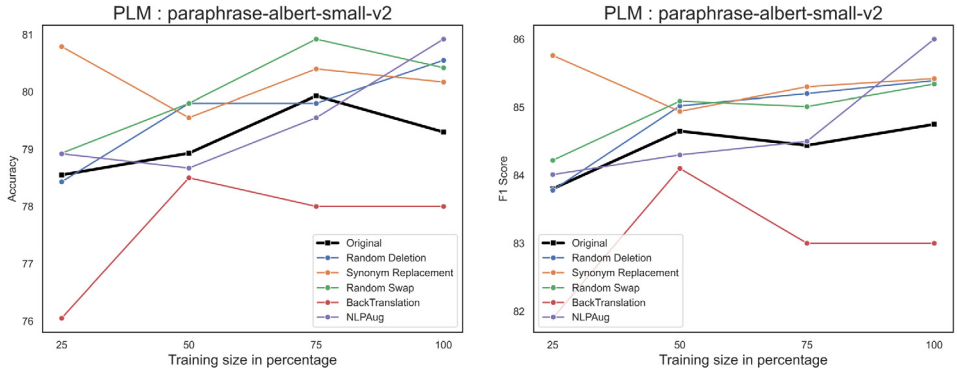
[8] https://github.com/sridevibonthu/SPRAG.

[9] https://nlpaug.readthedocs.io/en/latest/augmenter/char/ocr.html.



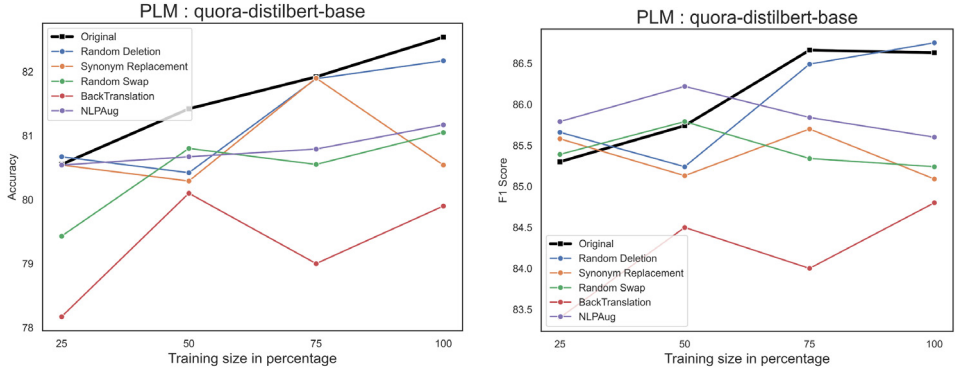
(a) Accuracy (b) F1-Score

Fig. 2. Performance of the five adopted augmentation strategies with different training sizes on the *stsb-distilbert-base* pretrained language model.



(a) Accuracy (b) F1-Score

Fig. 3. Performance of the five adopted augmentation strategies with different training sizes on the *paraphrase-albert-small-v2* pretrained language model.



(a) Accuracy (b) F1-Score

Fig. 4. Performance of the five adopted augmentation strategies with different training sizes on the *quora-distilbert-base* pretrained language model.

Table 3 presents the accuracy and F1-score on test data of all the trained models. The first row in every table is the performance of the sentence-transformer fine-tuned with the SPRAG dataset alone. The performance of the models trained on the training data is better (1%–3%) compared with the models trained with 25%, 50%, and 75% of the training data. It states the impact of the amount of training data on the models. The second row presents the performance of the models trained with gold (SPRAG) data and silver data obtained after randomly applying all the augmentations present in AugFS by following the algorithm 1. It is evident that the application, all in all, has not given any performance boost. To further identify the effect of all augmentations, every technique is applied individually. The rest of the columns from 3 to 7 present the results of those. The addition of silver data to gold data with individual application of augmentation while fine-tuning the model has given performance gain for all techniques except BT and NA. The increase in accuracy is $2-3\%$ and the F1-score is $1-2\%$ under both low-resource and high-resource settings. The plots of accuracy and F1-scores of these individual experiments are shown in the Figs. 2, 3, and 4.

A single outlier can prevent the underlying association in Pearson correlation calculation and is sensitive to the slightest departures from normality (Zhelezniak et al., 2019). Spearman is a robust correlation coefficient and it is just a Pearson's coefficient between ranked

Table 4
The spearman correlation rank (ρ) on the 84 experiments.

Data	25%	50%	75%	100%
	ρ	ρ	ρ	ρ
stsb-distilbert-base				
Original	0.58	0.62	0.62	0.64
AAug	0.58	0.61	0.63	0.62
RD	0.56	0.6	0.62	0.62
SR	0.57	0.61	0.62	0.63
RS	0.56	0.59	0.59	0.59
BT	0.57	0.6	0.62	0.61
NA	0.56	0.6	0.59	0.6
paraphrase-albert-small-v2				
Original	0.57	0.58	0.6	0.6
AAug	0.57	0.59	0.59	0.6
RD	0.58	0.58	0.61	0.61
SR	0.58	0.59	0.61	0.62
RS	0.53	57.6	0.57	0.57
BT	0.57	0.57	0.59	0.62
NA	0.57	0.58	0.57	0.6
quora-distilbert-base				
Original	0.6	0.62	0.64	0.64
AAug	0.62	0.61	0.63	0.64
RD	0.6	0.6	0.63	0.63
SR	0.6	0.62	0.62	0.64
RS	0.58	0.59	0.59	0.61
BT	0.59	0.6	0.61	0.62
NA	0.6	0.6	0.59	0.63

variables. As Pearson correlation is not well suited to the STS benchmark (Reimers et al., 2016), the spearman rank correlation is computed based on the similarity of the embeddings and the true labels with the help of sentence-transformers[10] library. The Table 4 presents the spearman rank correlations obtained for 84 models on test data. There is no improvement in the spearman rank of the trained models, as augmentation changes the sentences, thereby embeddings. The conclusion from the comparison of accuracies and F1-scores (Figs. 2, 3, 4) is that RS, SR, and RD are good augmenters; NLPAug is an average augmenter; and finally, BT is a worse augmenter for these models.

6.1. Ablation study

Tables 3 and 4 demonstrate the empirical results of the augmentations on pre-trained sentence transformers. In order to get a better understanding of the effectiveness of augmentation, an ablation study was performed with the observed good augmenters and the results are shown in the Table 5. Different combinations of RS, SR, and RD are evaluated on the entire training data. The combination of RD+RS+SR has shown significant improvement with the *stsb-distilbert-base model*, with a rise of 1.59% in accuracy and 1.63% in F1 score. But RD+RS+SR has shown significant impact on the other two models due to the change it is causing in the distribution of the data. All of the combinations resulted in a performance gain for the *para-albert-small-v2*. The combination of RS and SR has given an excellent rise of 4.91% in accuracy and a 3.36% rise in F1-score. The classification error on this model with the $RS + SR$ combination is 0.157. On the validation data, the confidence interval for the classification error with 90% probability is 0.157 ± 0.021. Therefore, the generalizability of the best model trained with $RS + SR$ on the *para-albert-small-v2* model has a 90% likelihood that the confidence interval $[0.135, 0.178]$ covers the true classification error on the validation data. The model *quora-distilbert-base* did not show a significant improvement with augmentations as it is already

[10] https://github.com/UKPLab/sentencetransformers/blob/master/sentencetransformers/evaluation/EmbeddingSimilarityEvaluator.py.

Table 5
Ablation results for good augmenters (RS, RD, SR) of our method on SPRAG dataset. The combination of RS and SR contribute significantly.

AugFs applied	Accuracy	F1-Score
stsb-distilbert-base		
No Aug.	81.30	85.63
RD + RS + SR	82.89	87.26
RD + RS	81.56	86.54
RD + SR	81.39	86.37
RS + SR	81.89	86.72
paraphrase-albert-small-v2		
No Aug.	79.3	84.75
RD + RS + SR	81.72	86.41
RD + RS	82.55	86.27
RD + SR	81.39	86.37
RS + SR	**84.21**	**88.11**
quora-distilbert-base		
No Aug.	82.54	86.63
RD + RS + SR	81.72	85.99
RD + RS	82.05	86.56
RD + SR	81.06	85.41
RS + SR	82.56	86.80

pre-trained with a huge quora question pair dataset.[11] In the ASAG setting, the pre-training is more on the classification task as the domain knowledge itself is enough to find the similarity. Fine-tuning improved the model's generalizability on the target dataset. The application of augmentations further enhanced the model.

7. Conclusion

This work proposes a method for scoring programming-related responses by leveraging both transfer learning and augmentation techniques. The proposed method mainly experiments with fine-tuning the pretrained sentence transformers and the usage of augmentation on generalizability when trained with both limited and full data as well. The experimental results on quora-distilbert-base indicate that the domain knowledge of PLMs plays a crucial role in the generalizability of the model. Fine-tuning the model enhances the training, and augmentation helps the model further enhance its performance. The combination of RS and SR has shown good generalizability on the SPRAG dataset. This work can be extended further in the following ways: (1) developing text augmentation strategies that do not alter the meaning of the sentence; (2) generating more synthetic data with the help of text GANs; and (3) python programming domain adaptation by enhancing the pre-training by using python-related books and webpages.

CRediT authorship contribution statement

Sridevi Bonthu: Conceptualization, Methodology, Software, Visualization, Writing – original draft. **S. Rama Sree:** Supervision, Validation, Writing – review & editing. **M.H.M. Krishna Prasad:** Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data is publicly available

[11] https://www.kaggle.com/c/quora-question-pairs.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Bayer, M., Kaufhold, M.-A., Reuter, C., 2022. A survey on data augmentation for text classification. ACM Comput. Surv. 55 (7), 1–39.

Bonthu, S., Dayal, A., Lakshmi, M., Rama Sree, S., 2022. Effective text augmentation strategy for nlp models. In: Proceedings of Third International Conference on Sustainable Computing. Springer, pp. 521–531.

Bonthu, S., Rama Sree, S., Krishna Prasad, M., 2021. Automated short answer grading using deep learning: A survey. In: International Cross-Domain Conference for Machine Learning and Knowledge Extraction. Springer, pp. 61–78.

Burrows, S., Gurevych, I., Stein, B., 2015. The eras and trends of automatic short answer grading. Int. J. Artif. Intell. Educ. 25 (1), 60–117.

Chapelle, C.A., Chung, Y.-R., 2010. The promise of NLP and speech processing technologies in language assessment. Lang. Test. 27 (3), 301–315.

Condor, A., Litster, M., Pardos, Z., 2021. Automatic Short Answer Grading with SBERT on Out-of-Sample Questions. International Educational Data Mining Society.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A., 2017. Supervised learning of universal sentence representations from natural language inference data. arXiv preprint arXiv:1705.02364.

Dai, X., Adel, H., 2020. An analysis of simple data augmentation for named entity recognition. arXiv preprint arXiv:2010.11683.

Dong, X., Shen, J., 2018. Triplet loss in siamese network for object tracking. In: Proceedings of the European Conference on Computer Vision. ECCV, pp. 459–474.

Dzikovska, M.O., Nielsen, R.D., Brew, C., Leacock, C., Giampiccolo, D., Bentivogli, L., Clark, P., Dagan, I., Dang, H.T., 2013. Semeval-2013 Task 7: The Joint Student Response Analysis and 8th Recognizing Textual Entailment Challenge. Technical Report, North Texas State University Denton.

Gao, F., Zhu, J., Wu, L., Xia, Y., Qin, T., Cheng, X., Zhou, W., Liu, T.-Y., 2019. Soft contextual data augmentation for neural machine translation. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 5539–5544.

Grangier, D., Iter, D., 2021. The trade-offs of domain adaptation for neural language models. arXiv preprint arXiv:2109.10274.

Heilman, M., Madnani, N., 2013. ETS: Domain adaptation and stacking for short answer scoring. In: Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013). pp. 275–279.

Jia, R., Liang, P., 2016. Data recombination for neural semantic parsing. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 12–22.

Jimenez, S., Becerra, C., Gelbukh, A., 2013. SOFTCARDINALITY: hierarchical text overlap for student response analysis. In: Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013). pp. 280–284.

Kumar, S., Chakrabarti, S., Roy, S., 2017. Earth mover's distance pooling over siamese LSTMs for automatic short answer grading.. In: IJCAI. pp. 2046–2052.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R., 2019. Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.

Loshchilov, I., Hutter, F., 2018. Fixing weight decay regularization in adam.

Lun, J., Zhu, J., Tang, Y., Yang, M., 2020. Multiple data augmentation strategies for improving performance on automatic short answer scoring. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. pp. 13389–13396.

Miller, G.A., 1995. WordNet: a lexical database for english. Commun. ACM 38 (11), 39–41.

Mohler, M., Bunescu, R., Mihalcea, R., 2011. 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.

Reimers, N., Beyer, P., Gurevych, I., 2016. Task-oriented intrinsic evaluation of semantic textual similarity. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. pp. 87–96.

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

Riordan, B., Horbach, A., Cahill, A., Zesch, T., Lee, C., 2017. Investigating neural architectures for short answer scoring. In: Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications. pp. 159–168.

Ruder, S., Peters, M.E., Swayamdipta, S., Wolf, T., 2019. Transfer learning in natural language processing. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials. pp. 15–18.

Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

Sennrich, R., Haddow, B., Birch, A., 2015. Improving neural machine translation models with monolingual data. arXiv preprint arXiv:1511.06709.

Silfverberg, M., Wiemerslage, A., Liu, L., Mao, L.J., 2017. Data augmentation for morphological reinflection. In: Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection. pp. 90–99.

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

Tan, C., Wei, F., Wang, W., Lv, W., Zhou, M., 2018. Multiway attention networks for modeling sentence pairs. In: IJCAI. pp. 4411–4417.

Taylor, L., Nitschke, G., 2018. Improving deep learning with generic data augmentation. In: 2018 IEEE Symposium Series on Computational Intelligence. SSCI, IEEE, pp. 1542–1547.

Wang, W.Y., Yang, D., 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 2557–2563.

Wei, J., Zou, K., 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196.

Wilianto, D., Girsang, A.S., 2023. Automatic short answer grading on high school's E-learning using semantic similarity methods.

Xia, Y., Qin, T., Chen, W., Bian, J., Yu, N., Liu, T.-Y., 2017. Dual supervised learning. In: International Conference on Machine Learning. PMLR, pp. 3789–3798.

Zhang, M., Baral, S., Heffernan, N., Lan, A., 2022. Automatic short math answer grading via in-context meta-learning. arXiv preprint arXiv:2205.15219.

Zhang, X., Zhao, J., LeCun, Y., 2015. Character-level convolutional networks for text classification. Adv. Neural Inf. Process. Syst. 28.

Zhao, S., Zhang, Y., Xiong, X., Botelho, A., Heffernan, N., 2017. A memory-augmented neural model for automated grading. In: Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale. pp. 189–192.

Zhelezniak, V., Savkov, A., Shen, A., Hammerla, N.Y., 2019. Correlation coefficients and semantic textual similarity. arXiv preprint arXiv:1905.07790.