International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

CNN-integrated NLP methods for Automatic Grading of Student Programming Assignment

Sidhidatri Nayak[1*], Reshu Agarwal[1], Sunil Kumar Khatri[2], Masoud Mohammadian[3]

Submitted:10/03/2024 **Revised**: 25/04/2024 **Accepted**: 02/05/2024

Abstract: Automatic grading has recently acquired significant traction in the educational industry due to its ability to revolutionize the evaluation process. With the advent of communications technology, automatic grading has grown quickly in the educational sector. Manually assessing and grading programming assignments is a time-consuming and demanding undertaking. Furthermore, the results of manual review are more subject to human mistake. Hence, there is a need to automate the grading process using modern techniques such as machine learning and Natural Language Processing (NLP). This study describes an innovative strategy to automating the grading process that combines Convolutional Neural Networks (CNN), Natural Language Processing (NLP), and transformation techniques including BERT, ROBERTA, and DistilBERT. The suggested approach takes advantage of CNN's ability to extract important information from both the assignment and the grades automatically. These attributes are subsequently translated using NLP and transformation models, resulting in a single representation of the assignments' quality, accuracy, and clarity. The CNN model's performance is evaluated using transformation algorithms for Unigram, Bigram, and N-gram text. The results of the experiment show that CNN-based NLP transformation approaches routinely beat traditional methods of computerized grading.

Keywords: *Automatic Grading, Natural Language Processing, Convolutional Neural Networks, BERT, ROBERTA, DistilBERT, Automatic Evaluation*

1. Introduction

In the digital age, as the education sector is witnessing huge transformation educators are facing challenges in terms of evaluating student programming assignments. With the surge in the submission of programming assignments, it becomes challenging for the academicians to evaluate the assignments and provide grading [1][2]. The increasing programming assignments also results in the unprecedented burden on educators, who must meticulously review and grade each assignment, often consuming substantial time and resources [3]. Automatic grading is one of the important aspects of the evaluation process [4][5][6]. With the emergence of communications technology automatic grading has been rapidly expanding in the educational sector. Conven tional approaches for grading depend on the evaluation of a student's assignments. In such approaches, handwritten assignments are the primary medium for evaluation. However, handwritten scripts incorporate certain challenges such as: evaluation of handwritten scripts is time consuming, and it is difficult to maintain consistency during evaluation. Besides there are high chances of error occurrences during evaluation because of manual intervention [7]. This problem can be addressed using advanced machine learning and deep learning technologies for evaluating the assignments and providing accurate grades [8][9][10]. However, these also suffer from certain drawbacks in terms of text recognition, and classification. In such cases, techniques such as Natural Language Processing (NLP)[11] are employed for interpreting the text accurately from the handwritten scripts and thereby help in evaluating the texts automatically. NLP has gained vast significance among various researchers because of its essentiality in automatic evaluation of handwritten scripts in examination processes [12]. However, handwritten assignments consist of diverse writing patterns and the similarity of characters can make it difficult to provide accurate grading. Hence it is essential to develop a high-reliability approach which identifies different types of handwritten characters, which is a challenging task.

Recently, transformation techniques such as BERT, ROBERTA, DistilBERT models are extensively used in applications such as text classification, aspect-based sentiment analysis, emotion recognition etc. [13][14]. These models represent the unlabeled textural data in a structured form for classification. This research focuses on developing a potential approach for providing automatic grading to the student programming assignments using CNN, NLP and transformation techniques. The study intends to employ efficient feature extraction techniques for enhancing the accuracy of the grading process.

[1]*Amity Institute of Information Technology, Amity University, Noida, India*
[2] *Research, Innovation and Extension Activities. Amity University, Noida, India*
[3]*University of Canberra, Canberra, Australia*
* *Corresponding Author Email: sidhidatri.nayak@gmail.com, ragarwal1@amity.edu, skkhatri@amity.edu, masoudm991@gmail.com*

The contributions of the work are as follows:

● This paper presents a novel approach for automatic grading that combines CNN with transformation models and NLP. Multiple text processing techniques such as tokenization, lemmatization etc. are used for data cleaning to improve the quality of the assignment text.

● A dependency graph is created using NLP transformation for extracting the text data. The text is clustered using a Term Frequency-Inverse Document Frequency (TF-IDF) and K-means clustering techniques, which generates labeled clusters for the CNN for classification.

● The text is further split into Unigram, Bigram, Trigram, and N-gram for enabling accurate grading of the assignment and the performance of the CNN is tested for the BERT, ROBERTA, DistilBERT models.

Section II reviews existing literary works on the automatic grading of the student's assignment. Section III discusses the proposed CNN and NLP-based approach for automatic grading and text classification. Section IV presents the results and Section V outlines the conclusion.

2. Literature Review

Various researchers have discussed the implementation of automated learning-based techniques such as ML and DL algorithms for automatic evaluation and grading of student programming assignments. This section discusses existing literary works done on smart evaluation and grading of programming assignments. The authors in [15] provided a comprehensive analysis of the ML and DL models for computerized assessment. These models explore the mapping of text related features for interpreting the automated essay scores. A deep convolutional neural network (DCNN) approach was presented in [16] for predicting the student's score. The DCNN classified the score of different students based on the characteristics of the text image rather than focusing on text-based methods after recognizing text. Outcome of the experimental analysis shows that the DCNN approach achieves an accuracy of 85% for a 10-score error and 95% for an error of 20 scores. A heuristic and ML model is implemented for the automatic evaluation of student's performance [17]. Five ML models were evaluated for identifying the quality of the submission and predicting the grades. Results show that the NB classifier achieves excellent accuracy of 91.07% in terms of other models compared to the manual grading system. Huang [18], employed a ML model for designing an intelligent scoring system. Experimental evaluation shows that the proposed ML based scoring system achieved an accuracy of 91.5% and reduced the cost of labor and minimized the time taken for evaluation. Several research works have used NLP [19] [20] and pre-trained transformer models such as BERT, ROBERTA and DistilBERT model for designing an automatic grading system. The authors presented an analysis of the n-gram text representation which is widely employed in automatic grading systems. Three n-gram representation techniques were employed for denoting the terms such as unigram, bigram and N-gram. A binary weighting methodology was applied for every answer script document with cosine similarity for comparing the answer scripts of different students. The effectiveness of this method was evaluated by comparing it with real time university datasets and results show that this technique can be implemented for automated grading systems across various universities. A linear regression and transformer model-based approach is presented in for short answer grading. Different combinations of BERT DistilBERT - RoBERTa models were used for grading the answers. Results show that the BERT model achieved an excellent performance in terms of grading which is trained on the English dataset.

It can be inferred from the literature review that existing works employ various single methodologies for different processes for evaluating the handwritten answer scripts. However, these methodologies provided unsatisfactory results. This is considered as one of the important limitations of automatic grading systems since it provides inaccurate results. Also, existing methodologies did not consider non-text contents such as figures, tables and mathematical expressions. Hence the need for developing an efficient automatic grading system is still persistent.

3. Proposed Research Methodology

This paper presents a novel and efficient and automatic grading system for evaluating student programming assignments using an integrated approach which combines CNN with NLP and transformer models. This research aims to classify the grades from the given data for realizing an accurate grading system. A CNN model is deployed as a classifier which is accompanied with NLP for interpreting and extracting textual data. The stages involved in the implementation of the CNN-NLP framework are discussed as follows: This paper presents a novel approach for automatic grading that combines CNN with transformation models and NLP. Multiple text processing techniques such as tokenization, lemmatization etc are used for data cleaning to improve the quality of the assignment text.

A dependency graph is created using NLP transformation for extracting the text data. The text is clustered using a Term Frequency-Inverse Document Frequency (TF-IDF) and K-means clustering techniques, which generates labeled clusters for the CNN for classification.

4. Data preprocessing

The data for the experimental analysis is collected from multiple student programs which are given as input to the CNN model and deployed the programming assignments as input datasets. In this research, multiple programs such as Fibonacci's series, swapping two numbers, factorial of a given number, square of a number, checking prime numbers, finding absolute value etc. are used as input. These programs are selected randomly and based on this a data frame is constructed with 500 programs. The dataset consists of texts which vary in size with millions of characters for complex analysis and modeling. The data frame is saved in the form of .CSV file. In this research, the datasets were not basically well-curated and preprocessed and thereby it is essential to clean and process the data to make them appropriate for the classification and analysis processes.

Preprocessing is crucial for automatic grading systems for various reasons. Preprocessing is done to make the data suitable by filtering out the redundant and unwanted information. In programming assignments, the data can be in various formats, programming languages, and coding styles. Data preprocessing can standardize the format and structure of submissions, ensuring that the grading system can work consistently across diverse submissions. Incomplete or missing data can be problematic for grading systems. Data preprocessing may involve strategies for dealing with missing data, such as imputation or ignoring instances with incomplete information. In this stage, the special characters in the text such as non-word characters (numbers), links, missing data, null data etc. are removed and only clean data is considered for the analysis. The concept of NLP is leveraged in this process which involves different steps such as tokenization, removal of stop words, stemming, lemmatization and PoS tagging, which are described in the following subsections:

4.1 Tokenization of text:

In this process, the input text is converted into a list of words (tokens) using a tokenize function known as NLTK. This function considers text input and divides the text into multiple words and then returns the list of tokens which assist the model to understand the text information and interpret the text accurately by accessing the words sequentially. Further, the tokenized input text is split and is stemmed using a Porter Stemmer function from NLTK (Porter Stemmer). The stemming process reduces the words to their basic root form and further returns the list of stemmed values.

4.2 Removal of Stop Words:

The words which are common English are removed from the input text. Stop words such as “and”, “the”, “is” etc are removed from the text since these words do not carry significant meaning in the text analysis process. The list of stop words are obtained from the NLT function which returns a list of tokens with stop words removed from the textual data. Removal of stop words helps in reducing the "noise" in the text, and thereby helps in the analysis of more meaningful words. In addition, removal of stop words reduces the number of unique words and makes the model more computationally efficient.

4.3 Lemmatization:

This lemmatization function lemmatizes the input text using the WordNet Lemmatizer function from NLTK. This process reduces the words considering the accurate meaning of the text. Lemmatization is more effective than stemming since it ensures that the lemmatized output consists of only valid words. These words are returned as a list of lemmatized tokens, which are further subjected for PoS tagging.

4.4 PoS tagging:

The part of speech (PoS) tagging uses the lemmatized tokens for tagging using the pos_tag function from the NLTK. In this process, the words or tokens are assigned with a grammatical label such as noun, verb, adjective etc. After tagging the words they are stored in the new columns of the data frame, and this helps in comparing and analyzing the impact of different text preprocessing techniques on the input data.

5. Dependency graph formation and NLP transformation

A dependency graph is a graphical representation that represents the grammatical relationships between words in a sentence or text. It is a fundamental concept in NLP used to analyze the syntactic structure of language and understand how words function within sentences. A sample illustration of the dependency graph is shown in figure.1 and the key elements of the dependency graph are discussed as follows:

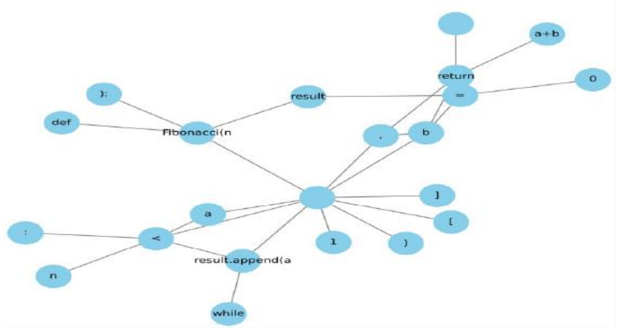

Fig.1. Sample dependency graph

The graph consists of prominent elements namely nodes, edges, and root nodes. In a dependency graph, each word or token in a sentence is represented as a node. These nodes contain information about the word such as its lemma, PoS tag, and other attributes. The nodes are connected by the edges which represent the grammatical relationships between words. Each edge has a label that describes the specific dependency or syntactic relationship between the connected words. Another important element is the root node which represents the main driving element (main verb) in the sentence. All other words in the sentence are connected to the root node directly or indirectly through a series of edges.

A feature extraction and text clustering technique based on TF-IDF vectorization along with KCM technique is employed in this research. The term Frequency represents the frequency of the word or phrase and IDF quantifies the significance of the words across the document. The TF-IDF helps in representing the text data as numerical vectors. In this research, the TF-IDF technique is employed for transforming unstructured text into a format that can be processed by the CNN model. It quantifies the importance of terms (words or phrases) in a document relative to a collection of documents (corpus). A "TfidfVectorizer ()" is created and is stored in the form of a vectorizer which transforms the text data into respective TF-IDF features. A TF-IDF score is obtained for each document which is measured by multiplying the TF and IDF values for a particular text. Each term in the document is associated with a dimension in the vector, and its TF-IDF score becomes the value in that dimension. The resulting TF-IDF vectors can be used as input features for the CNN model. These features are stored in the variable 'X'.

Further, a KCM algorithm is applied to the TF-IDF features stored in 'X' for creating multiple clusters. In this research 10 clusters are created. The KCM algorithm automatically determines the number of times the algorithm will be executed with different centroid seeds and selects the best result. Based on this, the cluster labels will be obtained from the K-means clustering and are stored in the `labels` variable. The words are concatenated, and each new text is separated by a newline character. It calculates and prints the total number of characters in the concatenated text data which is shown in figure 2.

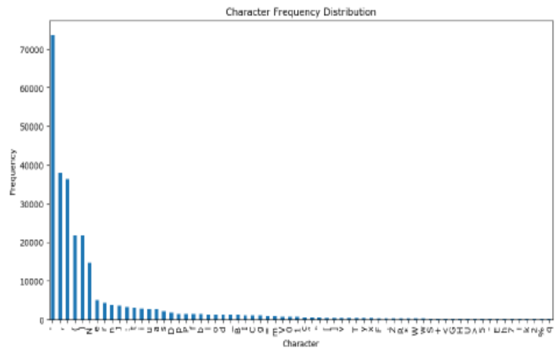

Fig.2. Character frequency distribution

As shown in figure 2, the KCM algorithm creates a bar plot to visualize the character frequency distribution that shows how often each character appears in the text. The `plot_wordcloud` function is called for each cluster separately, generating word clouds specific to the text data within each cluster. These word clouds help visualize the most common words within each cluster.

After forming the clusters, the text is further split into Unigram, Bi-gram and N-gram which is classified using CNN model with an embedding of transformer models namely BERT, DistilBERt and ROBERTa.

6. Classification using CNN model

CNN models are extensively used in the classification and text recognition tasks because of its ability to extract meaningful features from the layers automatically. In this research, a one-dimensional CNN (1D-CNN) is used for text classification.The CNN model is built with a BERT model, CNN with DistilBERT and CNN with ROBERTa model for feature extraction and classification for Uni-gram, Bi-gram and N-gram Texts. BERT (Bidirectional Encoder Representations from Transformers) is a NLP tool which has a transformer architecture for handling sequential data. BERT tokenizer input text into subword tokens using a WordPiece tokenizer. This allows BERT to handle out-of-vocabulary words and capture meaningful subword representations. DistilBERT is a variant of the BERT that retains much of the knowledge and capabilities of the original BERT model while significantly reducing its size and computational requirements. The architecture of the DistilBERT is the same as that of the BERT model with multiple layers of transformers for capturing contextual information from text. On the other hand, RoBERTa (A Robustly Optimized BERT Pretraining Approach) is a highly optimized variant of the BERT which is trained on a large corpus of publicly available text. However, RoBERTa uses substantially more data and longer training times, leading to improved language understanding. The performance of these models with CNN will be discussed in the next section.

7. Experimental Results

The CNN model is determined with different transformer models namely BERT, DistilBERT and ROBERTa for classifying Uni-gram, Bi-gram and N-gram texts. The performance of the CNN is measured with respect to different performance metrics. It is essential that an efficient requirement classification model must have a high classification accuracy with minimum loss function.

7.1 Performance of CNN with BERT:

The performance of CNN with BERT for classifying Uni-gram, Bi-gram and N-gram are discussed as follows: The confusion matrix of the proposed CNN classifier with BERT are shown in below figures. For experimental analysis the data was split into a ratio of 80:20 and the model is trained using both training and testing data.

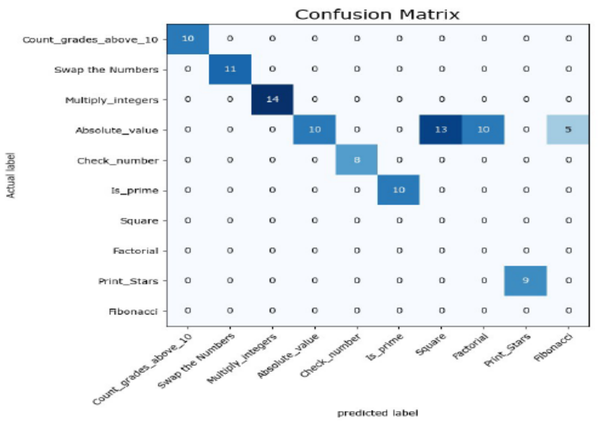

Fig.3. Confusion matrix of the CNN-BERT model for unigram text

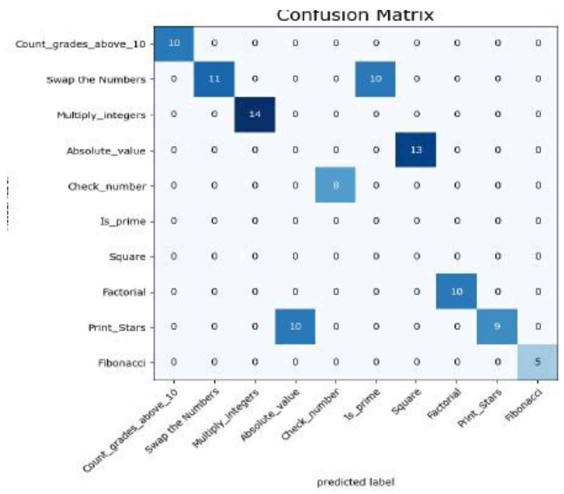

Fig.4. Confusion matrix of the CNN-BERT for Bi-gram text

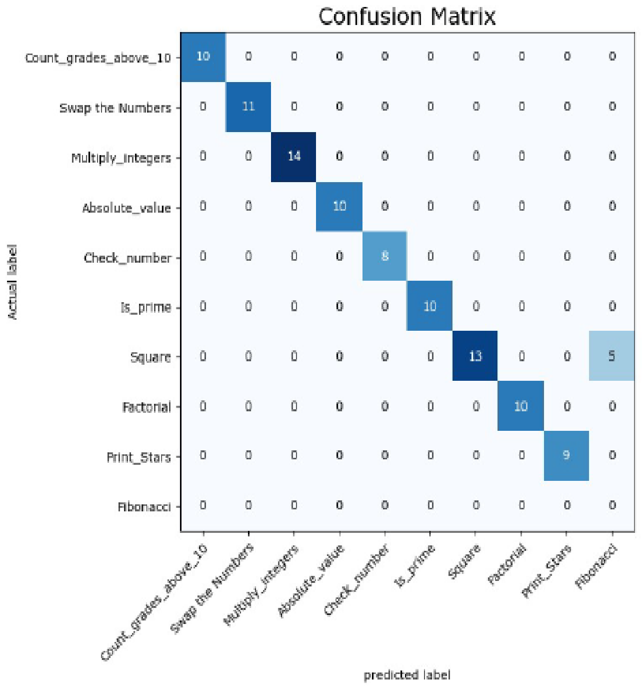

Fig 5. Confusion matrix of the CNN-BERT for N-gram text

The proposed approach achieves a moderate accuracy of 72 % with a highest precision of 64 %, recall of 72 %, sensitivity of 62.63 %, specificity of 97.12 %, F1 score of 66.16 %, and a false alarm rate (FAR) = 0.028.

The performance of the CNN model is shown in figure 6.

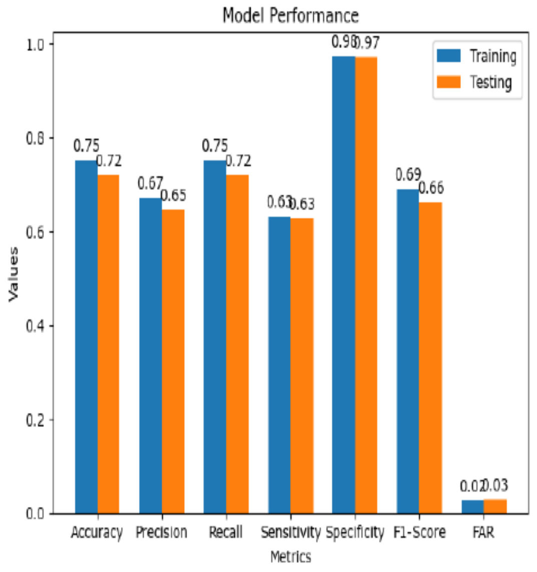

Fig 6. Performance of the CNN-BERT model for unigram text

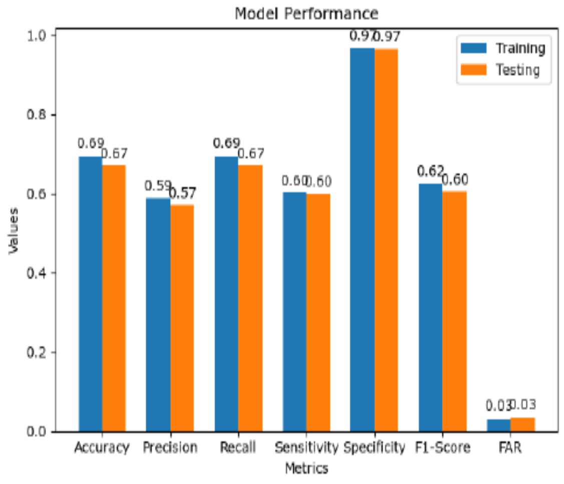

Fig 7. Performance of the CNN-BERT model for Bi-gram text

The proposed approach achieves an accuracy of 95 % with a highest precision of 87.65 %, recall of 92 %, sensitivity of 85.42 %, specificity of 99.19 %, F1 score of 89.18 %, and a false alarm rate (FAR) = 0.008.

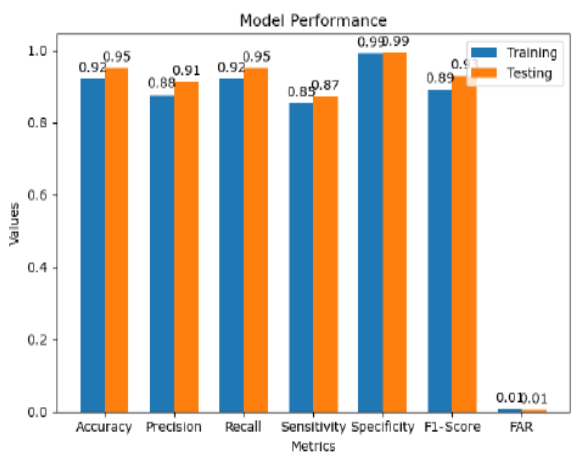

Fig 8. Performance of the CNN-BERT model for N-gram text

7.2 Performance of CNN with DistilBERT:

The performance of CNN with DistilBERT for classifying Uni-gram, Bi-gram and N-gram are discussed as follows.

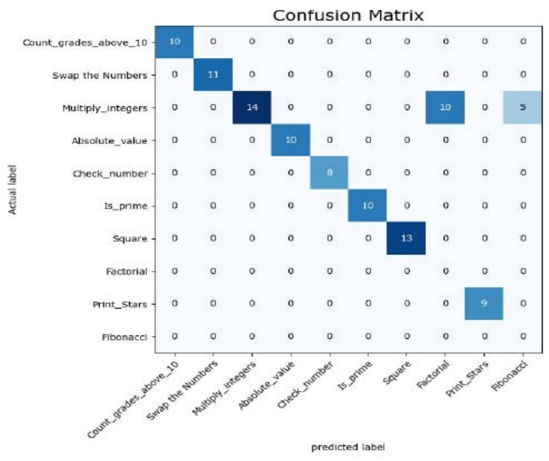

Fig 9. Confusion matrix of the CNN-DistilBERT model for unigram text

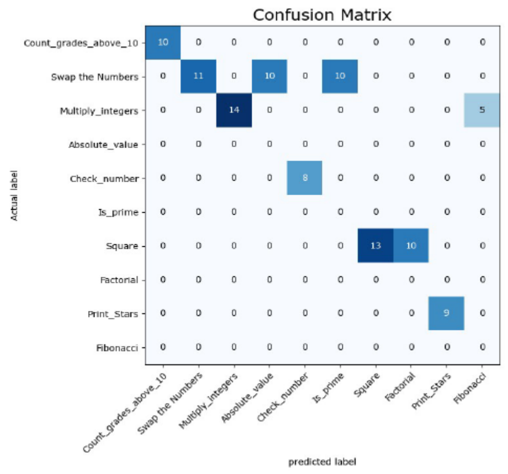

Fig 10. Confusion matrix of the CNN-DistilBERT for Bi-gram text

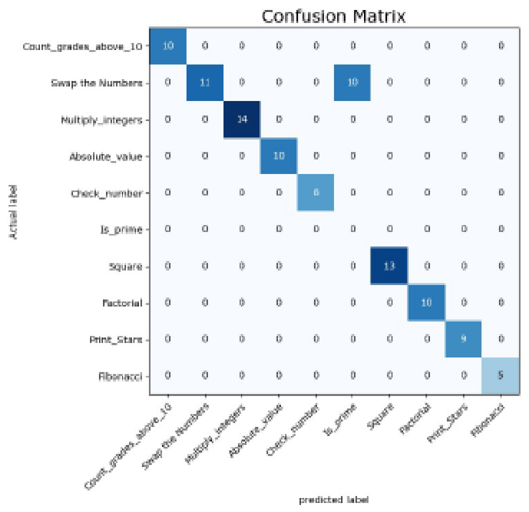

Fig 11. Confusion matrix of the CNN-DistilBERT for N-gram text

The proposed approach achieves a moderate accuracy of 85 % with a highest precision of 78.6 %, recall of 84.75 %, sensitivity of 74.0 %, and a false alarm rate (FAR) = 0.015. .

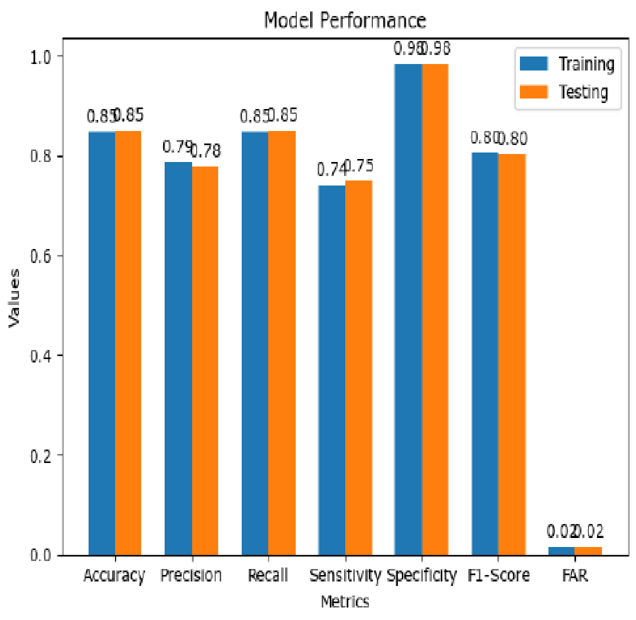

Fig 12. Performance of the CNN-DistilBERT for unigram text

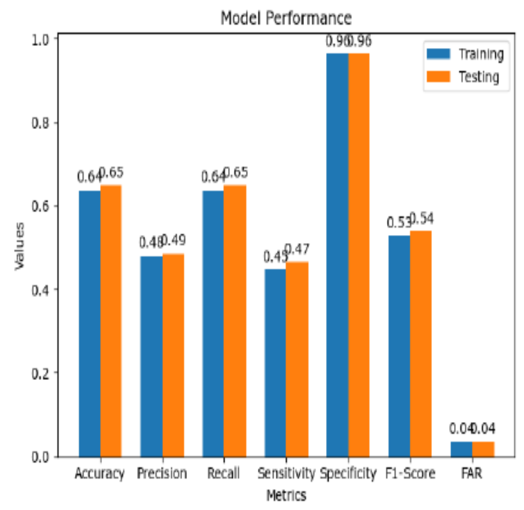

Fig 13. Performance of the CNN-DistilBERT model for Bi-gram text

The proposed approach achieves an accuracy of 90 % with a FAR of 0.01.

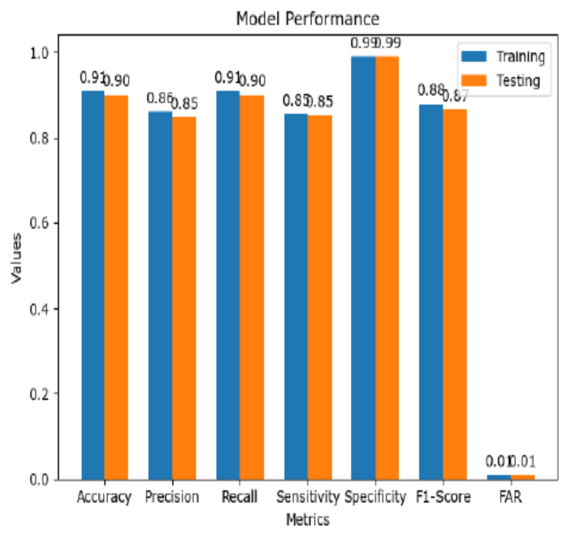

Fig 14. Performance of the CNN-DistilBERT model for N-gram text

7.3 Performance of CNN with ROBERTA:

The performance of CNN with ROBERTa for classifying Uni-gram, Bi-gram and N-gram are discussed as follows.

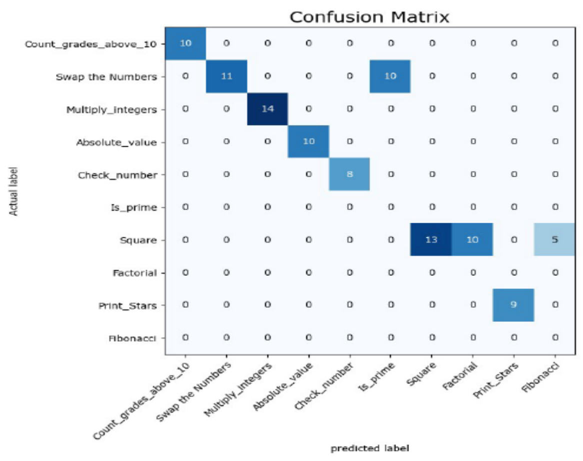

Fig 15. Confusion matrix of the CNN-ROBERTa for unigram text

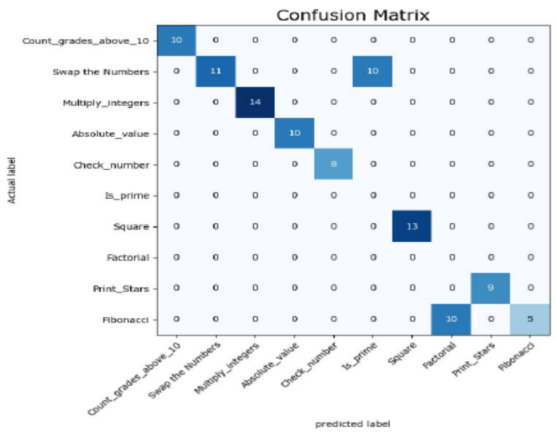

Fig 16. Confusion matrix of the CNN-CNN-ROBERTa for Bi-gram text

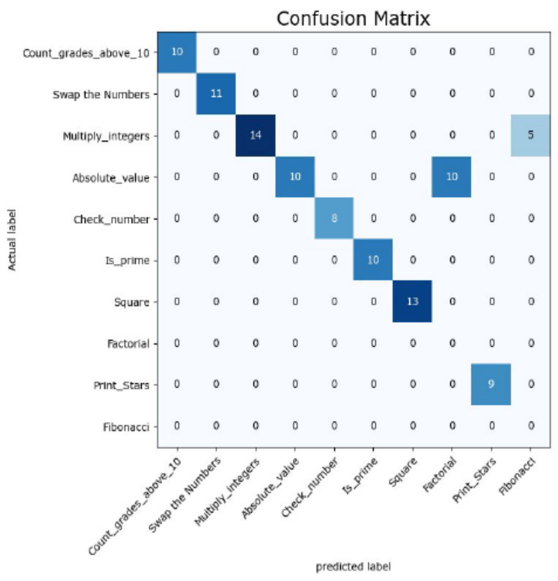

Fig 17. Confusion matrix of the CNN-ROBERTa for N-gram text

The proposed approach achieves a moderate accuracy of 75 % with a highest precision of 64.99 %, recall of 75.75 %, sensitivity of 59.28 %, specificity of 97.57 %, and a FAR score of 0.024.

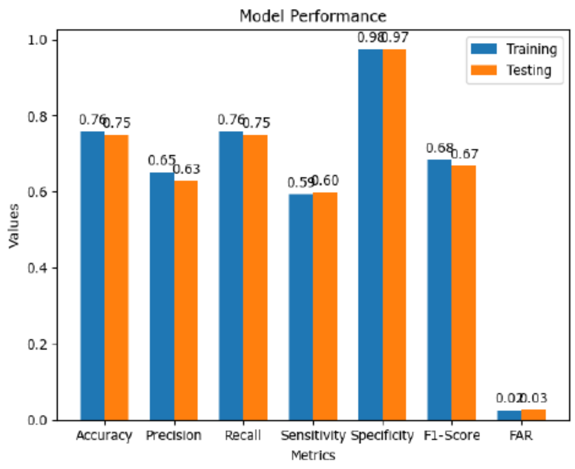

Fig 18. Performance of the CNN-ROBERTa model for unigram text

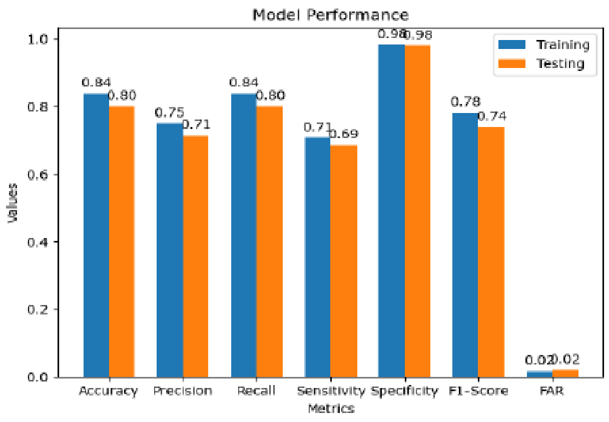

Fig 19. Performance of the CNN-ROBERTa model for Bi-gram text

The proposed approach achieves an accuracy of 85 % with a highest precision of 76.3 %, recall of 85 %, and sensitivity of 72.36 %, with a FAR of 0.015.

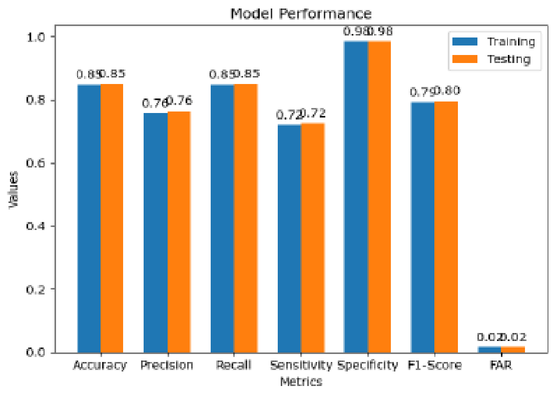

Fig 20. Performance of the CNN-ROBERTa model for N-gram text

8. Comparative Analysis

The performance of the BERT, ROBERTa, and DistillBERT models for unigram, Bi-gram, and N-gram text are as discussed in Table 1, 2 and 3 respectively.

Table 1. Comparison of different models for Unigram

Metrics	BERT	ROBERTa	DistillBERT
Accuracy	82 %	82 %	93 %
Precision	73 %	75 %	89 %
Recall	82 %	82 %	93 %
Sensitivity	72 %	74 %	86 %
Specificity	98 %	98 %	99 %
F1 score	76 %	77 %	91 %
FAR	0.02	0.02	0.01

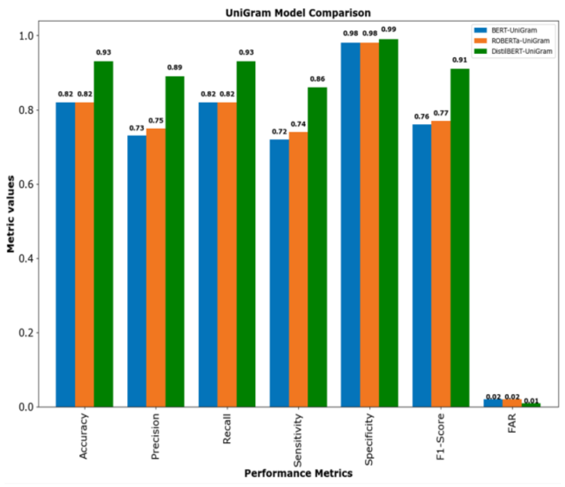

Fig 21. Performance comparison of three different models for Unigram text

Table 2. Comparison of different models for Bi-gram

Metrics	BERT	ROBERTa	DistillBERT
Accuracy	85 %	92 %	93 %
Precision	76 %	87 %	89 %
Recall	85 %	92 %	93 %
Sensitivity	72 %	86 %	86 %
Specificity	98 %	99 %	99 %
F1 score	79 %	88 %	90 %
FAR	0.02	0.01	0.00

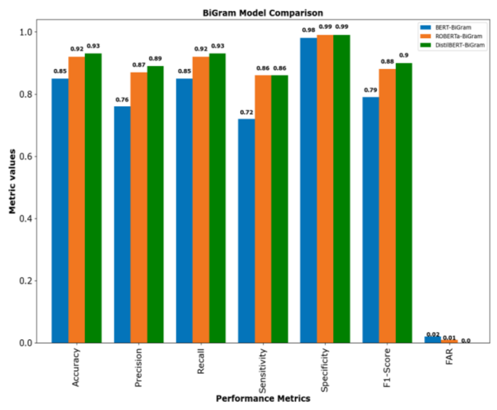

Fig 22. Performance comparison of three different models for Bigram text

Table 3. Comparison of different models for N-gram

Metrics	BERT	ROBERTa	DistillBERT
Accuracy	92 %	92 %	93 %
Precision	87 %	86 %	89 %
Recall	92 %	92 %	93 %
Sensitivity	86 %	86 %	86 %
Specificity	99 %	99 %	99 %
F1 score	89 %	88 %	90 %
FAR	0.01	0.01	0.00

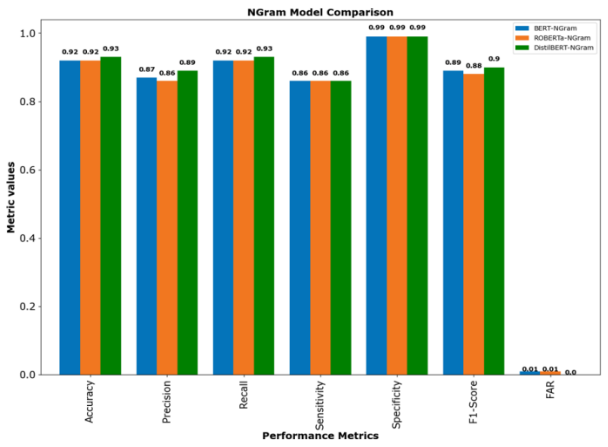

Fig 23. Performance comparison of three different models for N-gram text

The comparative results state that the FAR score of all the three models are very lower and among them, the model exhibited a least FAR score for N-gram text. The highest accuracy was also obtained for N-gram text wherein the accuracies exhibited by the BERT, ROBERTa, and DISTILLBERT models are 92 %, 92 % and 93 % respectively.

9. Conclusion

This paper discussed the performance of the CNN model for providing accurate grading for student's assignment automatically. The proposed approach consisted of efficient tools such as NLP and transformer models such as BERT, ROBERTa, and DistilBERT for accurately classifying the grades. The effectiveness of the CNN is determined with all three transformer models for classifying Unigram, Bi-gram and N-gram texts and is validated from simulation results where it exhibits better results. These parameters validate the efficiency and potential capability of CNN. A comparative analysis is conducted wherein the models were compared with respect to different evaluation metrics for three different types of texts. Results demonstrate that the CNN-based NLP transformation techniques consistently outperform traditional methods of automatic grading in terms of achieving a least FAR score and high accuracy. Moreover, the model exhibits strong generalization capabilities, achieving accurate grading even for previously unseen assignments.

References

[1] Liu, X., Wang, S., Wang, P., & Wu, D. (2019, May). Automatic grading of programming assignments: an approach based on formal semantics. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET) (pp. 126-137). IEEE.

[2] Wang, Z., Liu, J., & Dong, R. (2018, November). Intelligent auto-grading system. In 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS) (pp. 430-435). IEEE.

[3] Aldriye, H., Alkhalaf, A., & Alkhalaf, M. (2019). Automated grading systems for programming assignments: A literature review. *International Journal of Advanced Computer Science and Applications*, *10*(3).

[4] Nabil, R., Mohamed, N. E., Mahdy, A., Nader, K., Essam, S., & Eliwa, E. (2021, May). Evalseer: an intelligent gamified system for programming assignments assessment. In *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)* (pp. 235-242). IEEE.

[5] Tianyi, S., Yulin, K., Yihong, H., & Yujuan, Q. (2019, May). PAAA: An implementation of programming assignments automatic assessing system. In *Proceedings of the 2019 4th International Conference on Distance Education and Learning* (pp. 68-72).

[6] Gordillo, A. (2019). Effect of an instructor-centered tool for automatic assessment of programming assignments on students' perceptions and performance. *Sustainability*, *11*(20), 5568.

[7] Abdul Salam, M., El-Fatah, M. A., & Hassan, N. F. (2022). Automatic grading for Arabic short answer questions using optimized deep learning model. *Plos one*, *17*(8), e0272269.

[8] Ahmed, A., Joorabchi, A., & Hayes, M. J. (2022). On Deep Learning Approaches to Automated Assessment: Strategies for Short Answer Grading. *CSEDU (2)*, 85-94.

[9] Borade, J. G., & Netak, L. D. (2021). Automated grading of essays: a review. In Intelligent Human Computer Interaction: 12th International Conference, IHCI 2020, Daegu, South Korea, November 24–26, 2020, Proceedings, Part I 12 (pp. 238-249). Springer

International Publishing.

[10] Hernández-Blanco, A., Herrera-Flores, B., Tomás, D., & Navarro-Colorado, B. (2019). A systematic review of deep learning approaches to educational data mining. *Complexity*, *2019*.

[11] Rokade, A., Patil, B., Rajani, S., Revandkar, S., & Shedge, R. (2018, April). Automated grading system using natural language processing. In *2018 Second international conference on inventive communication and computational technologies (ICICCT)* (pp. 1123-1127). IEEE.

[12] Mello, R. F., Neto, R., Fiorentino, G., Alves, G., Arêdes, V., Silva, J. V. G. F., ... & Gašević, D. (2022, September). Enhancing instructors' capability to assess open-response using natural language processing and learning analytics. In *European Conference on Technology Enhanced Learning* (pp. 102-115). Cham: Springer International Publishing.

[13] Shaheen, Z., Wohlgenannt, G., & Filtz, E. (2020). Large scale legal text classification using transformer models. *arXiv preprint arXiv:2010.12871*.

[14] Adoma, A. F., Henry, N. M., & Chen, W. (2020, December). Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition. In *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)* (pp. 117-121). IEEE.

[15] Jiao, H., He, Q., & Yao, L. (2023). Machine learning and deep learning in assessment. *Psychological Testing and Assessment Modeling*, *64*(1), 178-189.

[16] Yang, L. (2019). Automated Examination Grading Using Deep Learning Categorization Techniques (No. 857). EasyChair.

[17] Boubekeur, Y., Mussbacher, G., & McIntosh, S. (2020, October). Automatic assessment of students' software models using a simple heuristic and machine learning. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings* (pp. 1-10).

[18] Huang, Z. (2023). An intelligent scoring system for English writing based on artificial intelligence and machine learning. *International Journal of System Assurance Engineering and Management*, 1-8.

[19] Lan, A. S., Vats, D., Waters, A. E., & Baraniuk, R. G. (2015, March). Mathematical language processing: Automatic grading and feedback for open response mathematical questions. In *Proceedings of the second (2015) ACM conference on learning@ scale* (pp. 167-176).

[20] Shaik, T., Tao, X., Li, Y., Dann, C., McDonald, J., Redmond, P., & Galligan, L. (2022). A review of the trends and challenges in adopting natural language processing methods for education feedback analysis. *IEEE Access*, *10*, 56720-56739.