Grenze International Journal of Engineering and Technology, June Issue

Towards Automated Evaluation of Handwritten Theoretical and Programming Answer Scripts using OCR and NLP

Vibhakti Bagade[1], Kalyani Mutkure[2], Anushka Hande[3], Prajakta Aote[4], Aditya Lonkar[5] and Kalyani Mutkure[6]
[1,3,4,5]Computer Science & Engineering, Yeshwantrao Chavan College of Engineering, Nagpur, India
Email: {vibhakti.bagde, anushkahande3, prajaktaaote2003, adityalonkar197}@gmail.com
[2]Department of Community Physiotherapy, Datta Meghe College of Physiotherapy, Nagpur, India
Email: kalyanimutkure.948@gmail.com
[6]Department of Community Physiotherapy, Datta Meghe College of Physiotherapy, Nagpur

***Abstract*— This is a challenging task especially when we have to evaluate handwritten answer scripts consisting of theoretical and programming-based answers. This paper focuses on using Optical character recognition (OCR) and Natural Language Processing (NLP) to automate the grading process. Azure Computer Vision Read API achieved average cross-similarity of 91% over pretrained Tesseract OCR, which was only trained on 37% of manually written scripts, based on analysis of 150 answers. The scores from the evaluation framework were 2.56/5.0 and 3.55/8.0 for theoretical and programming answers respectively. Despite showing potential for automatic evaluation, there are difficulties with acknowledging varied hand-writing styles along with complicated programming method.**

***Index Terms*—Optical Character Recognition (OCR), Natural Language Processing (NLP), Answer Script Evaluation, Text Segmentation, Semantic Analysis, Code Assessment, Mixed Content.**

I. INTRODUCTION

One main core part of academic evaluation has been the assessment of handwritten answer scripts. Yet, the process still involves a fair amount of manual effort, which makes it tedious and prone to variation. As the scale of examinations increases, there is an increasing need for fair and efficient automated systems. Recent advancements in Artificial Intelligence (AI) such as Optical Character Recognition (OCR) and Natural Language Processing (NLP) have enabled us to translate handwritten text to a digital format efficiently [1][2]. But when a script has both theoretical answers and programming code, the automation also adds different layers of complexity.

Variations in handwritten scripts with mixed content pose considerable challenges for the state-of-the-art automated processing domain. The uniqueness of handwriting styles, semi-readable papers, and the complex structure of C or Python further leads to confusion when it comes to separating text and code. Either type of content can be evaluated by traditional systems and mixed content answer sheets cannot be evaluated. This limitation has hampered the reach of automated assessment systems.

Grenze ID: 01.GIJET.11.2.126_20

The objectives of this study were to evaluate the performance of Tesseract and Azure Computer Vision OCR tools in constructing an automated process to assess theoretical and programming answers, test the effectiveness of this system, and discuss areas of improvement to be considered in automated assessment systems.

II. Literature Review

Machine learning and artificial intelligence have greatly improved the evaluation of handwritten assessments and addressed the variation in evaluation of mixed theoretical and programming content. In this section, we discuss the current works of research in optical character recognition (OCR), natural language processing (NLP), and automated code assessment technique.

A. Advances in Handwritten Text Recognition

The state of the art in handwritten recognition systems had witnessed major advancements in the area of architecture of deep learning over the years. The integration of Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) has shown us outstanding performance in handling various handwriting styles [1]. Specifically, CNN-based architecture combined with Bidirectional LSTM (BiLSTM) has shown great performance on standard benchmarks by exploiting spatial and temporal features [2]. Specialized preprocessing techniques utilizing adaptive binarization and targeted noise removal has reported an accuracy boost of up to 18% [4].

Another advancement in this area are writer-adaptive frameworks (WAFs). The MetaHTR system, utilizing adaptive frameworks (WAFs). The MetaHTR system, utilizing 5-7% better than baseline models targeting unseen handwriting styles in training [15]. However, parsing theoretical text and programming code together has proven difficult because they have different structural properties.

B. Semantic Analysis Through NLP

Transformer-based architectures have revolutionized the domain of theoretical answer scoring. BERT based models, in particular, have demonstrated high effectiveness for both semantic similarity and paraphrased content [8]. Prior work has shown that performance can improve by regularizing the model to produce the same prediction for semantically equivalent inputs [9]. This is a huge step for the accurate grading of student responses that might say the same thing in different ways.

C. Code Assessment Automation

The automated assessment of handwritten programming code has specific technical problems and challenges which need specialized solutions. However, with more recent innovations that combine OCR with syntax-aware neural networks, it is now possible to reduce this error rate from 30% to 5% [11]. In modern systems, a three-stage process is commonly applied:

- Sample code-specific text recognition
- Error correction based on language model
- Detailed syntactical and execution validation

Context-aware evaluation frameworks are especially successful at evaluating code structure and programming style [12]. These systems are able to check not only syntactic correctness but also if a code respects specific coding standards, providing a more thorough evaluation of the abilities for programming by students.

D. Integration Challenges

Ongoing research in mixed-content assessment identifies some common challenges that still seem to linger:

- Identification and segmentation of content types
- Keeping the context across various types of content
- Uniform evaluation standards for all theoretical and practical aspects
- Scalability of the system while ensuring accuracy

These challenges continue to highlight the need for integrated approaches that can ensure both theoretical and programming content is covered while retaining quality and efficiency in the assessment process.

III. Methodology

A. Dataset Preparation

We collected a dataset of handwritten answer scripts from 75 undergraduate computer science students, where each student answered two different questions, one theoretical (Q1) and one programming-based (Q2) questions,

and studied 150 handwritten answers. Fig. 1 illustrates the workflow of the study proposed. Abode Scan was used to convert the answer sheets to high-quality JPEG images. Digitized images are preserved on A4 size (2480 x 3508 pixels) with an average file size of 2.3 MB, providing enough quality to ensure the right text recognition but not excessively high to be computationally complex [1][2].

B. Preprocessing Workflow

The preprocessing workflow started from image standardization, wherein, all the images were resized to the desired width (1024 pixels) while maintaining the original aspect ratio. We started enhancing that sequence by changing to grayscale with OpenCV's cv2. COLOR_BGR2GRAY function and then applying a Gaussian blur of 3x3 kernel size and $\sigma = 0.5$ to remove noise.

Finally, we performed adaptive thresholding with a block size and C value of 11 and 2. The last pre-processing implemented were morphology operations, dilation with size 2 and then erosion with another kernel size of 1 to further improve text clarity and remove artifacts [3][5].

C. OCR Implementation

We tested two separate OCR methods to determine their efficacy with handwritten text. It uses the Azure Computer Vision system set to API Version 3.2, with the confidence threshold set to 0.75, automatic language detection and layout analysis capabilities enabled. At the same time, Tesseract OCR has been implemented. The pre-processed images were then fed into both of the systems, with structured JSON with recognized text and confidence score as output [6][7][13].

D. Content Segmentation

Now the extracted text was split into 3 sections using Content segmentation: header, theoretical answers, and programming code snippets. Using regular expressions to identify coding syntax, this enabled the system to correctly classify code from text. Moreover, a standard header format was employed for proper segmentation of the meta information [11][14].

E. Evaluation Framework

An evaluation framework aimed to provide several evaluations for detailed performance descriptions. For OCR performance, we used the Word Error Rate (WER) defined as (S+D+I) / N, where S = substitutions, D = deletions, I = insertions and N = number of words. At the character level CER applied a similar formula. The textual similarities were computed based on the cosine similarity between both the TF-IDF vectors corresponding to the recognized and reference texts. Theoretical questions were scored based on semantic similarity from Sentence-BERT with a cosine similarity threshold of 0.75 and a weighted keyword match of 0.3 for answer assessment. Programming questions were marked by three main factors: syntax correctness (40% weight), execution result success (40% weight), and similarity comparison of code structures (20% weight) [8][9][17].

F. Statistical Analysis

The performance evaluation used strict statistical methods to validate results. For paired comparisons of OCR systems, we used paired t-tests, while relationships between similarity scores were evaluated with Pearson correlation coefficients. Differences between groups were analysed using ANOVA testing, the confidence interval being set at 95% (α=0.05), as a way to assure the statistical significance of the generated results [10].

IV. Results and Discussion

A. OCR Performance Analysis

As shown, the Azure Computer Vision outperformed Tesseract in every metric by a good margin, obtaining an average similarity score of 91% versus Tesseract's 37%. Azure scored 491 hits perfectly while Tesseract scored 0 hits perfectly with 98.78% superior performance in other cases of lower WER (Word Error Rate) CER (Character Error Rate). These findings corroborate recent studies emphasizing in-Cloud OCR systems benefits [1][2][16]. Fig. 2 and Table I illustrates the OCR performance analysis.

B. Answer Evaluation Results

The descriptive analysis showed that the students' performance has different patterns on the theoretical (Q1) and the programming (Q2) question. For theoretical answers (Q1), there was a mean score of 2.56/5.0 (SD = 1.96), ranging from -0.45 to 5.49. (N (Q1) = 553); the distribution of Q1 marks was bimodal (Fig. 3), implying two

separate performance clusters for students. This bimodality is possibly a symptom of differences in theoretical understanding or epistemic approach to theoretical questions.

Answers on programming (Q2) were higher on average (M = 3.55/8.0, SD = 1.84), with scores between 0 and 7.34. Fig. 3 represents a right-skewed distribution in Q2, indicating that although most students received moderate scores, a few did exceptionally well. Both the higher mean (CV = 0.52 vs 0.76 for Q1) and lower coefficient of variation indicate more consistent performance when participants were asked to do programming tasks as compared to theoretical questions.

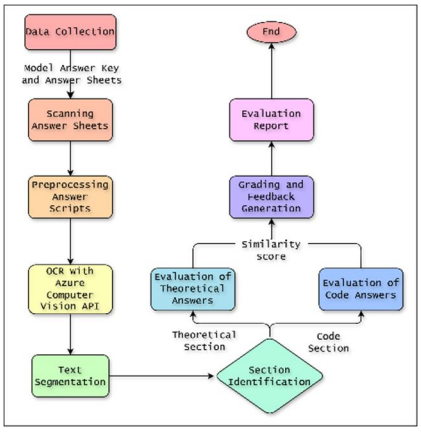

Figure 1. Proposed Workflow of Automated Evaluation

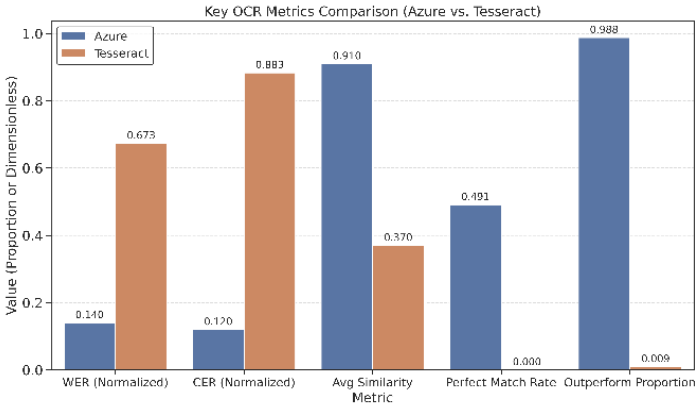

Figure 2. Comparison of OCR Performance

TABLE I. PERFORMANCE SUMMARY OF AZURE AND TESSERACT

Metric	Count	Mean	Std. Dev	Min	25%	Median	75%	Max
Azure Similarity	1389	0.91	0.165	0.00	0.899	0.970	1.00	1.00
Azure WER	1389	0.42	0.750	0.00	0.00	0.20	0.50	10.00
Azure CER	1389	0.36	2.540	0.00	0.00	0.044	0.14	53.00
Tesseract Similarity	1389	0.37	0.179	0.00	0.259	0.333	0.456	0.93
Tesseract WER	1389	2.02	2.405	0.33	1.00	1.17	2.00	31.00
Tesseract CER	1389	2.65	8.680	0.07	0.725	0.860	1.50	106.00

C. Key Statistical Findings

Our analysis reveals several major trends. Fig. 4 shows Q1 scores were strongly negatively correlated with Q2 scores ($r = -0.76$, $n=73$, $p < 0.001$) indicating that students with very good results in the theoretical questions did not necessarily perform well in programming tasks and vice versa. This unexpected relation opens the door to alternative paths of the study of student individual learning manners and basic evaluation methods.

D. System Limitations

Although promising, there are limitations that need consideration. The OCR system struggled with different handwriting styles, consistency of image quality, and recognized problems with special characters when converting the code to text.

This sometimes resulted in incorrect information that had to be checked by hand. While semantic similarity scoring is powerful, it is not always able to recognize statically similar, or contextually equivalent responses, that may not be directly matched at the word level such as unique complex theoretical answers.

Furthermore, the system is faint with its assessment of only partially correct code implementations, which hinders its contributive feedback nature. As context detection and assessment algorithms are not powerful enough.

V. CONCLUSION

They are one step towards the higher goal of achieving automated evaluation systems, but the results point to areas for improvement. That cloud-based OCR like Azure is way ahead of the game compared to traditional methods, is a good sign for handwriting recognition.

The framework produced accurate results through both theoretical and programming question evaluation; however, limitations remain, specifically with respect to handwriting style diversity and special characters in programming code.

FUTURE WORK

Future work should focus more on enhancing segmentation and handwriting recognition using advanced deep learning models, improved semantic analysis techniques as well as replacing the quality of syntax analysis for evaluating codes. Furthermore, widening the system to accommodate additional answer sheets would enhance its usability in multiple education institutions.

ACKNOWLEDGMENT

The authors thank Yeshwantrao Chavan College of Engineering for their support and for providing the facilities required for this study. Also deserve special thanks are those students, who provided their answer script written by hand to perform.

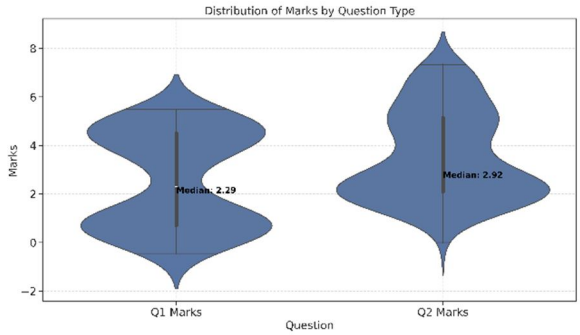

Figure 3. Distribution of Marks by Question Type

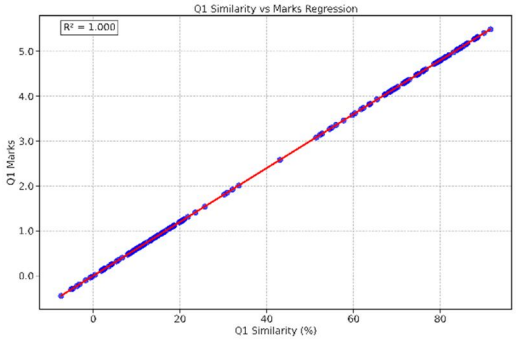

Figure 4. Q1 vs Q2 Score Correlation scatter plot

REFERENCES

[1] R. Chinthaginjala, C. Dhanamjayulu, and Th. Kim, "Enhancing handwritten text recognition accuracy with gated mechanisms," Scientific Reports, Volume 14, 16800 (2024). doi: 10.1038/s41598-024-67738-8.

[2] J. P. K. Kulkarni et al., "Enhanced Handwritten Text Recognition through Bidirectional LSTM and CNN Fusion," International Journal for Research in Applied Science and Engineering Technology, Volume 12, Issue 5, pages 3562–3569 (2024). doi: 10.22214/ijraset.2024.62233.

[3] J. Kohút, M. Hradiš, and M. Kišš, "Towards Writing Style Adaptation in Handwriting Recognition," Document Analysis and Recognition - ICDAR 2023, Lecture Notes in Computer Science, vol 14190, Springer, Cham, (2023). doi: 10.1007/978-3-031-41685-9_24.

[4] İ. Ç. Taş, and A. A. Müngen, "Using Pre-Processing Methods to Improve OCR Performances of Digital Historical Documents," Innovations in Intelligent Systems and Applications Conference (ASYU), Elazig, Turkey, 2021, pages 1–5. doi: 10.1109/ASYU52992.2021.9598972.

[5] R. Kaur, and D. V. Sharma, "Pre-processing for Improving the Recognition of Mobile-based Gurmukhi Text Recognition System," 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021, pages 1–5. doi: 10.1109/ICCCNT51525.2021.9579867.

[6] M. A. Sayeed, and D. Gupta, "Automate Descriptive Answer Grading using Reference-based Models," OITS International Conference on Information Technology (OCIT), Bhubaneswar, India, 2022, pages 262–267. doi: 10.1109/OCIT56763.2022.00057.

[7] R. Agrawal, H. Mishra, I. Kandasamy, S. R. Terni, and V. W.B., "Revolutionizing subjective assessments: A three-pronged comprehensive approach with NLP and deep learning," Expert Systems with Applications, Volume 239, 122470 (2024). doi: 10.1016/j.eswa.2023.122470.

[8] N. Srikanth, M. Carpuat, and R. Rudinger, "How Often Are Errors in Natural Language Reasoning Due to Paraphrastic Variability?" Transactions of the Association for Computational Linguistics, Volume 12, pages 1143–1162 (2024). doi: 10.1162/tacl_a_00692.

[9] Y. Li, H. Li, and J. Liu, "Towards Robust Neural Machine Reading Comprehension via Question Paraphrases," International Conference on Asian Language Processing (IALP), Shanghai, China, 2019, pages 290–295. doi: 10.1109/IALP48816.2019.9037673.

[10] E. Kacupaj, B. Banerjee, K. Singh, and J. Lehmann, "ParaQA: A Question Answering Dataset with Paraphrase Responses for Single-Turn Conversation," The Semantic Web. ESWC, Lecture Notes in Computer Science (LNISA), vol 12731, Springer, Cham, (2021). doi: 10.1007/978-3-030-77385-4_36.

[11] Md S. Islam, M. K. B. Doumbouya, C. D. Manning, and C. Piech, "Handwritten Code Recognition for Pen-and-Paper CS Education," Proceedings of the Eleventh ACM Conference on Learning @ Scale (L@S '24), Association for Computing Machinery, New York, NY, USA, 200–210 (2024). doi: 10.1145/3657604.3662027.

[12] L. Wehmeier, S. Eilermann, O. Niggemann, and A. Deuter, "Task-fidelity Assessment for Programming Tasks Using Semantic Code Analysis," IEEE Frontiers in Education Conference (FIE), College Station, TX, USA, pages 1–5 (2023). doi: 10.1109/FIE58773.2023.10342916.
[13] C. Sharp, J. van Assema, B. Yu, K. Zidane, and D. J. Malan, "An Open-Source, API-Based Framework for Assessing the Correctness of Code in CS50," Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'20), Association for Computing Machinery, New York, NY, USA, pages 487–492 (2020). doi: 10.1145/3341525.3387417.
[14] D. A. Ramos, and D. Engler, "Under-constrained Symbolic Execution: Correctness Checking for Real Code," Proceedings of the 24th USENIX Conference on Security Symposium (SEC'15), USENIX Association, USA, pages 49–64 (2015).'
[15] A. K. Bhunia et al., "MetaHTR: Towards Writer-Adaptive Handwritten Text Recognition," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, pages 15825–15834 (2021). doi: 10.1109/CVPR46437.2021.01557.
[16] S. Tobler, "Smart Grading: A Generative AI-Based Tool for Knowledge-Grounded Answer Evaluation in Educational Assessments," MethodsX, Volume 12, Article 102531 (December 2023). doi: 10.1016/j.mex.2023.102531.
[17] G. Reddy, "Enhancing Subjective Answer Evaluation Through Machine Learning and Natural Language Processing," International Journal of Scientific Research in Engineering and Management, Volume 08, Issue 03, pages 1–5 (2024). doi: 10.55041/ijsrem2952