ISSN 2319-5991 www.ijerst.com
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

SMART GRADER USING ADVANCEDNATURAL LANGUAGE PROCESSING FOR ACADEMIC ASSESSMENTS

P. Rakesh
PG Scholar, Department of Computer Science and Engineering, Jawaharlal Nehru Technological University Hyderabad University College Of Engineering Jagtial(Autonomous), Nachupally(Kondagattu)
Telangana, India - 505501
Email: perlarakesh037@gmail.com

Dr. G. Narsimha
Principal & Professor, Department of Computer Science And Engineering, Jawaharlal Nehru Technological University Hyderabad University College Of Engineering
Jagtial(Autonomous),Nachupally(kondagattu),
Telangana, India - 505501
Email: narsimha06@jntuh.ac.in

Abstract— Manual marking of descriptive handwritten tasks can be time-consuming and can cause inconsistent marking and late marking. This study presents an AI Assignment Grader which combines Optical Character Recognition, Natural Language Processing, semantic similarity, concept coverage, plagiarism analysis, automated grading, as well as AI-based feedback and report production in a role-based online application. Teachers design task and develop sample responses. Student answers are handwritten into pictures, PDFs or DOCX documents. PaddleOCR is a machine reading tool for uploading documents to obtain machine-readable texts. Using Sentence Transformer embeddings and cosine similarity, student responses are compared to reference answer . Marks, grades and notification of plagiarism are based on the concept coverage and similarity of submissions. The online application is powered by Flask, the academic data is stored in SQLite, PyMuPDF and Python-docx are used for the uploaded documents, and ReportLab is used to create downloadable academic evaluation reports. The project does not rely on a set dataset to learn a new classification model, rather the actual inputs are the instructor's answer key and student's assignments that are provided dynamically. Thus the implementation is evaluated based on the successful running of OCR, semantic analysis, plagiarism analysis, grading, feedback, analytics and report generation. The approach devised lowers the repetitious evaluation work, yet gives structured results and keeps the instructor in charge of the ultimate academic judgments.

Keywords— Automated grading, handwritten assignments, PaddleOCR, Sentence Transformers, cosine similarity, plagiarism detection, Flask.

I. INTRODUCTION

Descriptive assignments may be effective to assess conceptual understanding, reasoning and writing. In a traditional grading method, the teacher reads each answer, checks the concepts against the answers that are expected, marks the score and provides feedback. It is a very time-consuming process if the answers are handwritten answers that are several pages long and if uniformity across answers is difficult to obtain at times because of manual assessment.

Today the digital assessment techniques usually only involve multiple choice or written answers. The handwritten assignments pose further problems because the quality of the handwriting and the style of the handwriting, the orientation of the pages, the format of the documents, etc. make the extraction of text difficult. To be viable, an automated approach would need to first read the document submitted and convert it to a reliable language before assessing meaning – rather than just matching words to words.

The proposed AI Assignment Grader combines the capability of document processing and academic grading in a single online application. Assignments are created by teachers and reference answers posted. The students' answers are submitted in pictures, PDF or DOCX files. We use PaddleOCR for the extraction of handwritten text, Sentence Transformers for the generation of semantic representations, cosine similarity to assess contextual proximity and concept coverage to determine if essential concepts are covered. It also conducts plagiarism analysis, provides marks and grades, provides comments, saves results and creates downloadable reports.

This integration of these services in a role based Flask application for administrators, instructors and students is the contribution of this work. This solution uses a pre-trained OCR and language model, and no project specific classifier is trained nor is a predefined benchmark data set used. Therefore, the study provides the real input processing and system results without any dataset statistics and unjustified performance measures.

II. RELATED WORK

Rahman and Akter used NLP techniques for automated assessment of answer-scripts and Rani used machine-learning techniques for assessment of explanatory-answer. Nandini and Uma Maheswari [3] demonstrated the effectiveness of semantic relational features for the descriptive-answer score as compared to literal word matching. While these methods established the context analysis as an important way of approach, they focus more on scoring than the academic process itself.

Before the semantic interpretation of handwritten answers, the recognition of handwritten answers is necessary. Singh et al. [4] develop an automatic assessment method for subjective handwritten reply. In answer-sheet compositions, Wu et al. [5] examined text-line detection and recognition, while employing only a small amount of labelled data. Practical handwritten-text recognition from images was demonstrated by Mule et al.

[6]. Document-processing applications can effectively use OCR framework PP-OCR [10].

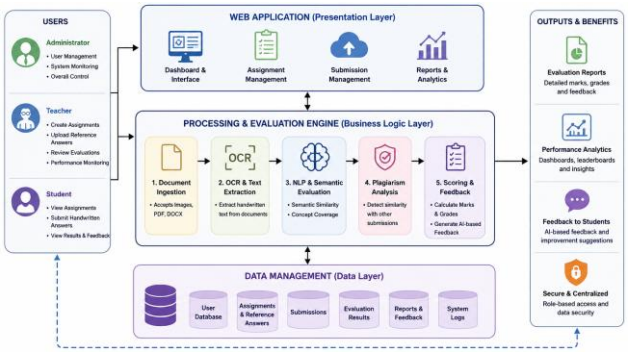
There have been notable developments in the area of short-text similarity approaches [12] and Universal Sentence Encoder [13] and BERT [14] which got very good contextual representation for sentence comparison. These tools allow to analyze comments in different phrasing and with different meaning. But semantic similarity is not the only way to assess ideas that are missing, plagiarism and academic comments. This proposed application combines OCR, phrase embeddings, idea coverage, plagiarism detection, scoring, role based management, analytics and report generation in a single platform.

Additionally, there was a diversity of automated grading systems that looked at different types of answers. Short response methods usually involve a comparison or contrast of one or two sentences, while descriptive methods can be paragraphs, examples and explanations. This is the first step in a semantic approach, and there can be a lot of mistakes in the handwritten input. Hence the final mark quality is subjective and depends on the extraction of the document and interpretation of the response. The instructor must be able to see any errors made by the student in the first step.

Another issue in the previous research is the detachment of a scoring system from the context in which it is going to be used, that of the school. A deployable system requires for it to have accounts, create assignments, maintain references and answers, validate files, track submission, store results, provide feedback, generate reports, and have instructor evaluation. The present project meets this requirement of integration without making unqualified assertions about the accuracy of the model, or about training a model.

III. MATERIALS AND METHODS

The proposed solution is in sync with the actual application architecture in the project. The system includes user authentication, assignment management, document processing, semantic evaluation, plagiarism detection, score generation, grade generation, feedback, report generation, analytics and ongoing database management. The overall system architecture is shown in Fig. 1.



"Fig. 1. Overall system architecture from the project document."

The architecture links the administrator, teacher and student interfaces to the Flask application server. Assignments are made by teachers and reference answers are kept. The student gets their answers written and it is handed over to the modules for further processing and assessment of the document. The marks, grade, plagiarism information, comments, notes and reports created are stored in SQLite and displayed via role-specific dashboards.

A) Input Description

The project does not have any specific training/testing data collection. Its inputs are automatically created by the academic workflow. The information about the assignment and the answers of the references is given by teachers. Students handwrite answers to assignments and upload to the website as pictures, PDFs or word documents. Student feedback and instructor feedback are also captured online in the system through forms. These inputs are checked before being processed.

The OCR module does the actual work of processing the images. PyMuPDF is used for reading PDF pages and pages are transformed to a form appropriate for recognition. Python-docx is used to manipulate DOCX files. PaddleOCR is used to obtain the text content of handwritten and scanned pages. The extracted text is cleaned for superfluous formatting elements and formatted for a semantic comparison against the instructor reference text.

The input-processing pipeline assigns to the correct assignment and student account each uploaded file. File-type checking – filters out documents that are not supported from the review process. Submission records are keeping upload time and state. The assignment stores the teacher model response, allowing the student responses for the same exercise to be evaluated in a similar context.

Evaluation inputs also include the max marks assigned to the assignment and the teacher directions. They also assist the feedback module to give feedback whether the answer is complete, partially complete or not yet. They define the range of the normalized score that is converted to marks. Comments are added by the teacher after automatic processing and added to the final report.

B) OCR and Text Pre-processing

Handwritten text submitted is converted to machine readable text using PaddleOCR. This step obviates the need for human transcription and is the primary source of data for Natural Language Processing. The retrieved text is checked to make sure it is complete and is "normalized"—that is, it is stripped of page breaks, extra spaces and other formatting artifacts to make it easier to process. If no text is detected and there is no meaningful text in the document, you must review the document or upload it again rather than using an auto score.

If the document is multi-page, the recognized text is collected in the sequence of the pages and the logical sequence of the answer is preserved. We remove unnecessary space, duplicate line breaks and conversion errors, but still keep the important punctuation and sentence structure as this relies on the surrounding words for contextual language models. So, the machine can assess the retrieved answer as a whole piece of academic writing and not as a single word.

There is still a chance for OCR to have some errors. Recognition quality may be decreased by poor lighting,

blurry pictures, very cursive writing, tables, equations and diagrams. Thus the program is designed as a teacher assistance program. The outcome achieved could be assessed and unclear work may be returned for re-submission. This approach will avoid the misinterpretation of an OCR failure as a lack of topic knowledge.

C) Semantic Similarity and Concept Coverage

We compute Sentence Transformer embeddings of the reference answer and the retrieved student response. Cosine similarity is used to measure their closeness in context:

$$S_sem = \frac{E_r \cdot E_s}{\|E_r\|_2 \, \|E_s\|_2} \quad (1)$$

where E_r and E_s refer to embedding of reference answer and student response, respectively. The higher the value, the more semantically similar the texts are. Concept Coverage percentage is the ratio of the ideas or concepts sent by the student to the ideas or concepts expected:

$$C_cov = \frac{\text{Number of covered concepts}}{\text{Total reference concepts}} \quad (2)$$

The score of the assessment is the result of the sum of the semantic similarity and the coverage of the concepts with adjustable weights:

$$S_base = \frac{\alpha S_sem + \beta C_cov}{\alpha + \beta} \quad (3)$$

With this type of pairing, it can also detect paraphrased responses, without losing important ideas.

Semantic similarity is the overall proximity of the context, whereas concept coverage is whether or not the ideas that are expected are present. Both the indicators complement each other. A good answer that doesn't fully address the question is likely to earn a good semantic score and not a full idea score; an answer composed of isolated words with no real logical reasoning should not be awarded full marks as an answer. These are then mixed together for first evaluation to give an organized experience.

Weights may vary according to type of assignment. In the case of concept-oriented questions, there may be a greater emphasis placed on coverage, while in explanatory questions there may be some flexibility in the area of semantic. The computed score is then normalised and converted to assignment marks and the teacher can comment on any alternative right logic used in the calculation.

D) Plagiarism, Marks, Grade, and Feedback

We have a plagiarism analyzer which analyzes the given response with other student responses and anticipates how similar the two are. The percentage that is reported is a marker for the teacher and will be incorporated into the final evaluation report. Citing from the conventional standard and required language is natural and not deemed as independent evidence of academic dishonesty.

The base score is then converted to marks, as outlined in the number of maximum marks awarded to the assignment. Application grading rules are used to assign a grade. Google Generative AI gives helpful feedback that leverages the extracted answer, the reference answer, the semantic evaluation, and missing concepts. They have the opportunity to view the product, give feedback and have control over the final academic assessment.

Student contributions are fed through a text extraction and pre-processing to avoid that formatting differences between the contributions will bias the similarity calculation. Result and report contain high percentage of . So, the teacher may review both the answers . . . . There is no automatic accusation, as there may be actual similarities in a query that is asking for a standard definition or a common technical word.

The feedback part is a translation of analytical indicators to human-readable instructions. It may recognize a concept that is covered, highlight a missing or inadequately explained concept, and offer suggestions to the student for improving the response. The feedback is for formative learning and is provided alongside marks, grade, plagiarism information and instructor remarks and not in lieu of teacher judgement.

E) Web Application and Report Generation

Flask comes with routing, session management, role-based access, handling uploads, and rendering templates. Users, assignments, submissions, evaluation results and reports are stored in a SQLite database. For PDF processing we use PyMuPDF, for image handling we use Pillow, for reading Word documents we use Python-docx, for similarity calculations we use Scikit-learn and for generating downloadable PDF assessment reports we use ReportLab. It has distinct dashboards for administrators, instructors and students to help them.

F) Functional Modules

User Authentication Module is responsible for Administrator, Instructor and Student registration, login, session and role based access. The Assignment Management Module allows the teacher to make and edit assignments, store reference answers, and track students' progress on assignment submissions. The Document Processing Module analyzes the content of uploaded Image, PDF and DOCX files, extracts them and prepares for evaluation a normalized text.

The Evaluation Module compares the extracted student response with the teacher reference answer, semantically checks the similarity of the content and the coverage of the topics, performs plagiarism analysis, calculates the marks and grade and provides AI-based feedback. The Report Generation Module creates a report that has a structured format, containing marks, grade, plagiarism details, student feedback and teacher comments. Analytics Module: It summarizes the submissions and performance. Database Management Module: This is responsible for managing users, assignments, submissions, results, reports, and academic information.

G) Deployment and Output Generation

The application is implemented with the deployment architecture based on the browser. The users communicate

with the Flask server using normal web pages. The Flask server performs the document processing, OCR, semantic analysis, plagiarism check, generation of the feedback, report preparation and database activities. The prototype uses the persistent storage provided by SQLite. The modular design allows the further evolution into a server database, which is suitable for institutional use.

The program provides scores , grades , results of semantic evaluation , information on plagiarism , AI-generated feedback , instructor comments , and a downloadable evaluation report . The teachers can view each submission and the progress of the class, while the students can view their submission history. The results are documented as part of the assignment documentation and the student record to make retrieval of the results easy without repeating the entire evaluation process.

The result is dependent on the hand-writing, image quality, file format, and if the OCR output is not empty and reference answers exist. It is required to develop input validation and error handling. The program should stop the assessment and request the correction of the input in case no text relevant to the assessment can be recovered. Teacher review is kept to rectify OCR mistakes and to check responses that include equations, diagrams, abbreviations or other acceptable explanations.

H) System Requirements and Security

It is development using Python 3.x, Flask, SQLite, PaddleOCR, Sentence Transformers, Scikit-learn, PyMuPDF, Python-docx, Pillow, ReportLab and Google Generative AI. Requires a current web browser and a regular personal computer. The prototype described does not use a dedicated graphics processor but if so desired, additional processing power would decrease OCR time and embedding time of massive submissions.

Role based authentication restricts access based on the roles of the user. Administrators can administer user accounts and control the entire application. Teachers can assign, save answer keys, review answers, analyze and provide feedback. Students can look at the tasks offered, submit their own replies and view just their own assessment scores. The rules on session management and the database are the basis of secure administration of academic records.

Implementing the assessment at the institutional level will require the use of encrypted communications, controlled filing of assessment documents, a strict password system, a backup system and audit monitoring. Feedback from external feedback providers should only be sent to the text required to create the feedback. The institution should define data retention and privacy policies prior to implementing it in production.

I) Feasibility and Practical Use

The chosen technologies are open source or low cost, and can be integrated without any particular hardware, which makes the application technically and economically viable. The lightweight web framework Flask, the document recognition model PaddleOCR, and the semantic representation models Sentence Transformers, without training a new language model. It is also in line with ordinary academic practice, with the role-oriented process also being practicable for assessment of assignments.

Usefulness of application is as a screening and feedback tool. It can remove the need for you to compare the content of the results, normalise the information for every result, and help teachers mark work for further assessment. There are certain things that can't be judged automatically, such as when the student's handwriting is hard to read, when the student responds imaginatively with an answer other than the one in the reference key, when the student responds with a picture, or when the student gives a legitimate answer but it is different from the answer in the reference key.

IV. IMPLEMENTATION AND RESULTS

A) Teacher Assignment Input

Teacher, set up an assignment by entering the title, description, marks, due date and referencing answer required for semantic assessment. Fig. The interface for creating an assignment, included in the project document, is displayed in Figure 2. Students are notified of the created task on their dashboard.

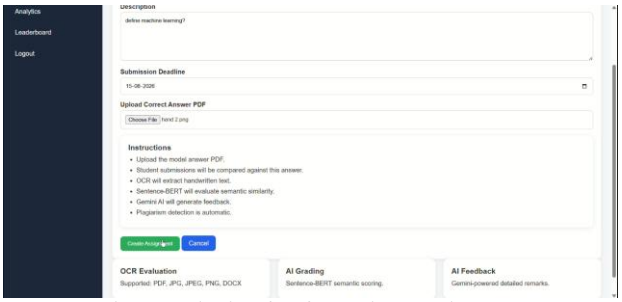


Fig. 2. Teacher interface for creating an assignment.

This is where the instructor can easily create assignments, collect student work, records of the assessment, leaderboard, and academic monitoring. The interface of project shown in Fig. 3 illustrates how the instructor has shifted from overseeing assignments to the automated review of assignments, in the role-specific interface.

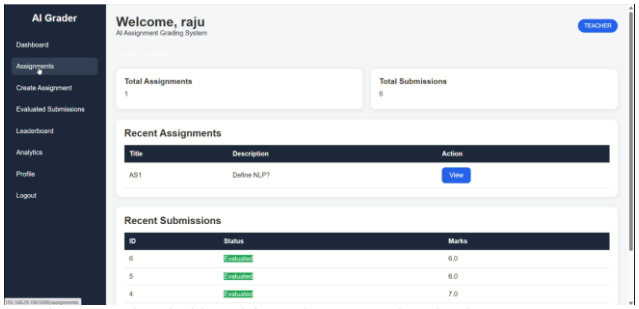


Fig. 3. Teacher dashboard for assignment and evaluation management.

B) Student Dashboard and Assignment Details

The student dashboard includes a list of available assignments, progress on assignments submitted, assessment scores, and student statistics. Fig. A dashboard is given in the project paper, which is displayed in Fig. 4. This page is where you can begin to select an assignment and review the details of that assignment.

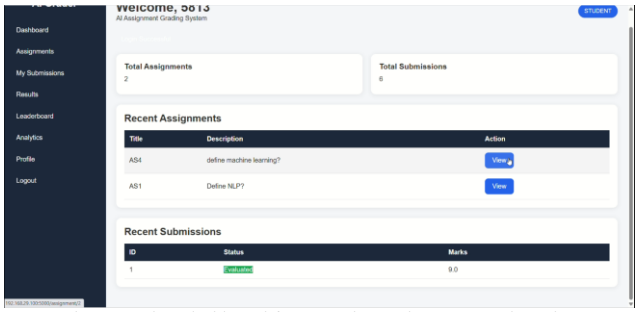


Fig. 4. Student dashboard for accessing assignments and results.

Students will first access the assignment details page on which the selected assignment and submission requirements are displayed. Fig. This project interface is shown in Figure 3. This guarantees that the answer is submitted in the relevant assignment before being passed to the assessment modules.

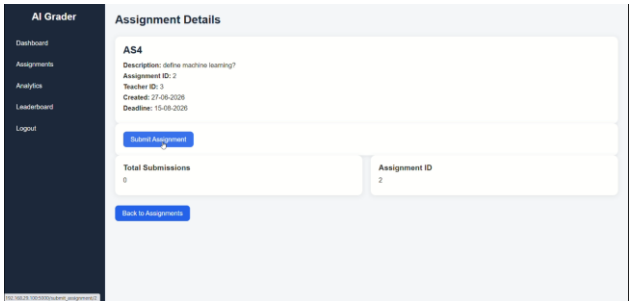


Fig. 5. Student assignment details page.

The student then uploads their answer in an accepted format. Fig. shows the submission interface. There's a direct input page to the automated assessment procedure at 4. The document submitted is checked, saved, and then OCRed and semantically analyzed.

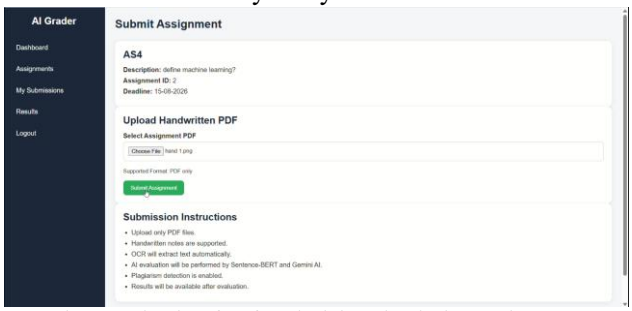


Fig. 6. Student interface for submitting a handwritten assignment.

C) Submission Confirmation and Teacher Review

On successful upload, the app records the upload and provides a confirmation that it is received. 1. The project document confirmation screen is shown in Fig. 5. Then, the teacher can view and assess the corresponding record.

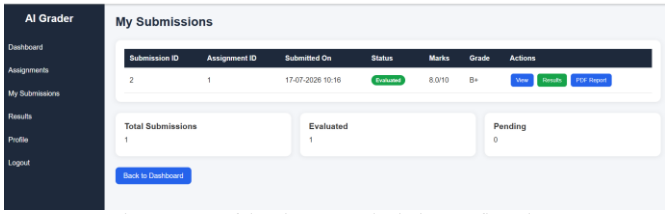


Fig. 7. Successful assignment submission confirmation.

Teachers can use the student-submissions interface to see submitted work and its assessment status. It connects each answer to the corresponding assignment, student record and provides access to the assessment generated. This interface is shown in Fig. . 6.

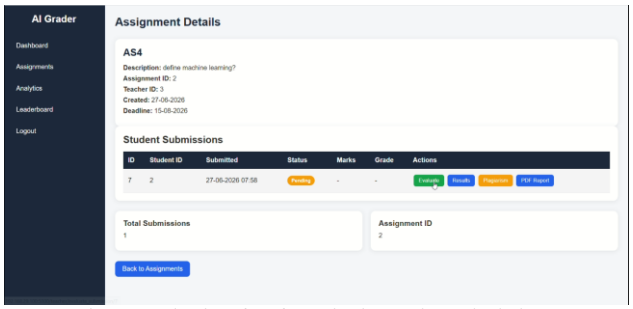


Fig. 8. Teacher interface for reviewing student submissions.

D) Evaluation Output

The main output of the application is the Evaluation Result screen. It displays successful marks, grade, semantic score, plagiarism, AI feedback and teacher notes as well as the report generation option. The outcome is therefore more descriptive than a letter grade and will enable it to be seen clearly by teachers in their review.

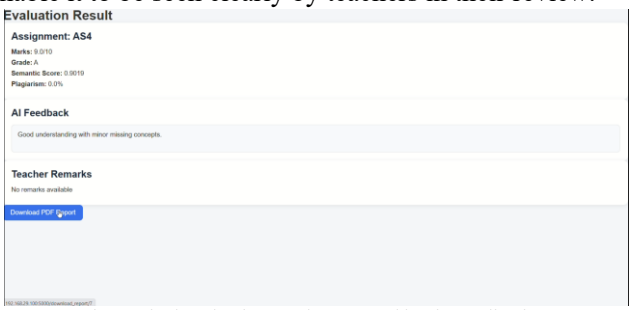


Fig. 9. Final evaluation result generated by the application.

E) Analytics and Performance Monitoring

The analytics feature provides you with a summary of assignments, submissions, grades and student performance. Fig. The project report includes a figure with the analytics dashboard which can be seen in Figure 8. These charts are neither application activity nor accuracy of the model, they are about the activities in the application. Since there is not a specific labelled dataset, Accuracy, Precision, Recall, F1-score and modelcomparison tables are not provided.

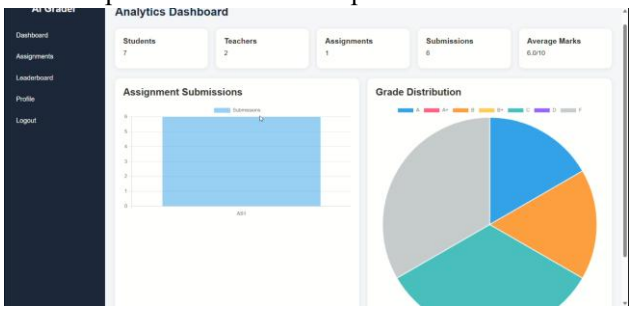


Fig. 10. Analytics dashboard for assignment and grade monitoring.

F) Result Analysis

The results of the output are consistent and reflect the prescribed academic procedure is in good working order. The app can make tasks and answers, students can upload handwritten papers, the software will extract text content, do semantic analysis, concept analysis, estimate plagiarism and calculate marks/grades, generate feedback and take the final result. The teachers will have access to the submissions and can write comments on it, and the students can view the result of the evaluation and download the report.

The implementation includes all essential modules mentioned in the project, including authentication,

assignment management, processing documents, assessment, report generation, analytics and database administration. The results confirm operation of integration and not the accuracy in making predictions from a newly trained model. This distinction can make a difference because PaddleOCR and Sentence Transformers are pretrained parts, and there is no project document defining a dataset, train-test split or expert labelled grading standard.

The approach minimizes the repetitive manual effort and offers consistent beginning signs but the final product should be checked by a teacher. Diagrams, mathematical equations, bad pictures and illegible handwriting can cause issues with OCR. Concept coverage and instructor remarks are also important components of the evaluation, since a topically relevant answer may lack some of the necessary components for a high score, but still receive a high score based on its similarity.

The resulting PDF report will be a record of the assessment. Combines the assignment and student information with the automated marks, grade, plagiarism mark, feedback and instructor comments. This report can be downloaded for students to reference and they can read the result without having to browse the numerous pages of the application.

The analytics site offers more general monitoring, by combining analysed data rather than revealing private answers. This information can be used by teachers to monitor grade distributions and submission and students to keep a track of their work in the role-specific dashboard. As a result, the online interface is connected to continuous performance monitoring, allowing for linkage to the individual assessment.

The main constraint is the lack of a validation corpus annotated by experts. There is no evidence that the modules are integrated when they are executed successfully, and no evidence that they produce the appropriate outputs (measured in terms of numerical scores), but there is evidence that they produce them. It should be used to make high-stakes decisions about student learning only in a controlled research before it is used in practice for students. The mark created should be considered as an explanation rather than a substitute for teacher assessment to that date.

V. CONCLUSION

This work introduces an AI Assignment Grader for automatic grade of the handwriting assignments which include the description of the handwriting. The software compiles all the above features into a Flask and SQLite application, including the following AI features: PaddleOCR, Sentence Transformer embedding, cosine similarity, idea coverage, plagiarism identification, automatic marks and grades, AI-based feedback, instructor comments, PDF report generation, and analytics. There are 3 roles in the system – Administrator, Teacher and Student, it supports image, PDF and DOCX submissions.

Students are not given any specific data set on which to base the project, but rather the answers of the students and the answers of the instructor are dynamically generated. The report therefore makes no unsupported claims of model performance measures. The developed interfaces and results demonstrate the successful implementation of the creation of lessons / assignments, the submission of documents, OCR processing, semantic processing, plagiarism detection, the grading of documents, feedback, reporting and performance monitoring.

Future work could include a controlled handwritten answer set of validated transcriptions, rubric ideas, independently monitored instructor marks and plagiarism labels. This data would help for quantitative assessment by OCR error rate, mark error, expert-store correlation, grade agreement and plagiarism precision and recall. Multi-language handwriting, equations, diagrams, multiple reference answers, integration with learning management systems and cloud deployment can further extend its applicability, and it is still possible to maintain the monitoring by the instructor.

A) Discussion and Practical Significance

The utility of the resulting system is that it integrates different types of activities which are typically performed separately. Handwritten pages don't need to be retyped, similarity can be measured by reading answers from top to bottom, repetitive comments are not required and a separate report is not required for each student. These processes are executed in the same assignment record in the program and the resulting indications are displayed in a uniform style.

Another aspect of the architecture is its ability to improve traceability. The saved result allows the teacher to have an assignment, student, submitted file, computed semantic score, percent plagiarized, marks, grade, feedback and notes. This is more informative than a standalone automatic mark, because the teacher can review supporting material and modify the concluding mark if handwriting recognition and/or semantic interpretation is inadequate.

The technique is especially beneficial for formative and internal evaluation since the speedy feedback allows students to improve before final tests. It can indicate missing ideas and give a standard first pass assessment throughout a big class. This enables the teacher to concentrate on the contentious answers to the alternate logic diagrams, and the subject-specific information that will need expert opinion. This division of labor avoids duplication of effort, yet not the intellectual responsibility by the teacher.

The reference and answer embedding can be stored to be scalable as all submissions to an assignment are embedded with the same reference. Multiple uploads can also be for OCR and semantic analysis background work. With the current assessment modules and the role based web interface, SQLite can be replaced with a ‘server' database and a central file storage for institutional use.

With ethical deployment, students should be aware they are being processed by automated means and should be able to seek review if the content that is being extracted is incorrect. Should be not to give final high stakes grades until confirmation from the teacher. Restrictions on privacy, limited access, secure storage and minimal disclosure of assignments and assessment reports to

external AI services is required because they are part of the student's academic records.

REFERENCES

[1] M. M. Rahman and F. Akter, "An automated approach for answer script evaluation using natural language processing," Int. J. Comput. Sci. Eng. Technol., vol. 9, pp. 39–47, 2019.

[2] A. M. Rani, "Automated explanatory answer evaluation using machine learning approach," Design Engineering, pp. 1181–1190, 2021.

[3] V. Nandini and P. Uma Maheswari, "Automatic assessment of descriptive answers in online examination system using semantic relational features," J. Supercomput., vol. 76, no. 6, pp. 4430–4448, 2020.

[4] S. Singh, Y. Shah, Y. Vajani, and S. Dholay, "Automated paper evaluation system for subjective handwritten answers," in Proc. IEEE ICCCNT, 2021, pp. 1–6.

[5] K. Wu, H. Fu, and W. Li, "Handwriting text-line detection and recognition in answer sheet composition with few labeled data," in Proc. IEEE ICSESS, 2020, pp. 129–132.

[6] H. Mule, N. Kadam, and D. Naik, "Handwritten text recognition from an image with Android application," in Proc. IEEE ICAECC, 2022, pp. 1–5.

[7] P. J. Sijimol and S. M. Varghese, "Handwritten short answer evaluation system," Int. J. Sci. Res. Sci. Technol., vol. 4, no. 2, pp. 1514–1518, 2018.

[8] S. Surana et al., "Text extraction and detection from images using machine learning techniques: A research review," in Proc. IEEE ICEARS, 2022, pp. 1201–1207.

[9] S. Pang et al., "A text similarity measurement based on semantic fingerprint of characteristic phrases," Chin. J. Electron., vol. 29, no. 2, pp. 233–241, 2020.

[10] Y. Du et al., "PP-OCR: A practical ultra lightweight OCR system," PaddlePaddle Technical Report, 2020.

[11] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. EMNLP-IJCNLP, 2019, pp. 3982–3992.

[12] T. Kenter and M. de Rijke, "Short text similarity with word embeddings," in Proc. ACM CIKM, 2015, pp. 1411–1420.

[13] J. Cer et al., "Universal sentence encoder," arXiv:1803.11175, 2018.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186.

[15] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. Pearson, 2023.

[16] T. Mikolov et al., "Distributed representations of words and phrases and their compositionality," in Adv. Neural Inf. Process. Syst., vol. 26, 2013.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[18] A. Vaswani et al., "Attention is all you need," in Adv. Neural Inf. Process. Syst., vol. 30, 2017.

[19] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[20] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[21] Y. Du et al., "PP-OCRv2: Bag of tricks for ultra lightweight OCR system," arXiv:2109.03144, 2021.

[22] Y. Du et al., "PP-OCRv3: More attempts for the improvement of ultra lightweight OCR system," arXiv:2206.03001, 2022.

[23] A. Graves, "Generating sequences with recurrent neural networks," arXiv:1308.0850, 2013.

[24] A. Bojanowski et al., "Enriching word vectors with subword information," Trans. Assoc. Comput. Linguistics, vol. 5, pp. 135–146, 2017.

[25] T. Brown et al., "Language models are few-shot learners," in Adv. Neural Inf. Process. Syst., vol. 33, pp. 1877–1901, 2020.

[26] P. Black and D. Wiliam, "Assessment and classroom learning," Assessment Educ., vol. 5, no. 1, pp. 7–74, 1998.

[27] D. Boud and N. Falchikov, Rethinking Assessment in Higher Education. London, U.K.: Routledge, 2007.

[28] J. Hattie and H. Timperley, "The power of feedback," Rev. Educ. Res., vol. 77, no. 1, pp. 81–112, 2007.

[29] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[30] A. Ronacher, "Flask: A Python microframework," Pallets Projects, 2010.