International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Automated Question Generation and Grading Using Artificial Intelligence

Manisha Mali[1], Rutvik Kachgunde[2], Ojas Deshpande[3], Srivatsa Gondas Ankamwar[4]

[1]Professor, Department of Computer Engineering VIIT, Pune Maharashtra, India.
[2,3,4]Department of Computer Engineering, VIIT, Pune Maharashtra, India

Abstract
In the current education system, the teachers and professors are responsible not only for imparting education but also for creating and conducting tests and giving students marks. Thus, causing them a tremendous amount of stress. Therefore, in this project, we are easing the professor's examination process. Simply how it works is, by giving input of study and reference material and asking the module to make questions. The second step would be, extracting the answers written by the student and then checking for keywords from the materials based on which the questions are formed and then comparing and giving marks as per the points in the answer.

Keywords: Artificial Intelligence, Question Formation, Answer Checking, Grading System, OCR

INTRODUCTION
Examinations are considered one of the most important in schools and colleges. However, the professors who need to curate the examinations, check the answers, and grade them are often ignored. As per the statistics, 84 percent of professors feel exhausted after a day of teaching. In such cases doing extra work like drafting question papers from scratch and proof checking them seems like a burden. More than 86 percent of students feel they aren't given the right marks, feel dissatisfied, and think they deserve more marks. Thus, building a relationship of distrust between the student and the professor. Currently, one of the things the professors do is, use the questions made previously in the past multiple years and use either a random selection function in Python or manually select a question and form the question paper. However, this method only helps ease the process of drafting the question paper. Yet, the papers need to be checked and graded manually. Checking the papers of a few students can be easy but if a professor needs to check thousands of papers, they get mentally and physically exhausted and could damage the fairness in checking the papers and thus, end up giving the candidate lesser marks than they deserve. Using the proposed methodology, we are not only easing out the process of question paper formation but also checking and grading the question papers. Where we just need to input the textbook or any document that is relevant to the course for which the exam is to be designed and the ML module will extract the materials given in the document i.e. will be done with the process of preprocessing. Once that is done, with the click of a button, we will get the questions generated for the students. After the questions are drafted and students are done taking the examination, the papers are scanned if the answer sheets are handwritten, and if the answer sheets are not handwritten, the file is added and then that data is put in as the input for the ML module

which checks the answer. The ML module is trained to search for keywords and once it does find the keywords, it will automatically give out grades to the students.

LITERATURE REVIEW

In Optical Character Recognition (OCR) PaddlePaddle OCR is a lightweight OCR system that consists of three parts, text detection, detection boxes rectification, and text recognition. It demonstrates great computational efficiency with the use of CPU rather than GPU considering the cost. The overall model size of PP-OCR is compact with only 3.5 MB for recognizing 6,622 Chinese Characters and 2.8 MB for recognizing 63 alphanumeric symbols. A series of strategies can be used to enhance the model capability. Large-scale datasets are used to train the system, which includes: 97K images for text detection, 600K images for direction classification, and 17.9M images for text recognition. This PP-OCR is also verified across multiple languages, such as Chinese, English, French, Korean, Japanese, and German, hence it is a versatile product. This PP-OCR model shows the balance between performance and size, making it suitable for real-world applications. **[1]**

The analysis of textual data is automatically required as the textual data lengthens; Convolutional Neural Networks (CNNs) show great effectiveness in it. The CNN architectures specifically designed for sentiment classification is one such approach, it demonstrated effectiveness for longer texts. The models used were evaluated on three well-known datasets and they achieved accuracies of 81% for binary classification and 68% for ternary classification. CNN's layered architecture improves performance, especially with deeper structures. **[2]**

Optical Character Recognition (OCR) has advanced a lot with many studies focusing on improving accuracy and efficiency across diverse text and image conditions. The research paper compares various OCR engines: Tesseract, Keras, PaddleOCR, and Microsoft Azure Computer Vision through various image preprocessing techniques: Edge Detection, Thresholding on datasets like IAM and FUNSD. Various metrics were used to evaluate the engines that includes: precision, Character Error Rate (CER), recall, Word Error Rate (WER) and processing speed. It showcases the importance of preprocessing techniques like Otsu's thresholding to enhance the performance of some engines. **[3]**

Language representation models have undergone significant evolution, with the emergence of pre-training techniques in Natural Language Processing (NLP). Training models like ELMo and GPT utilized unidirectional architecture. The model was constrained to process the input text either left to right or right to left. BERT (Bidirectional Encoder Representations from Transformers) is an approach that employs a deep bidirectional architecture that pre-trains on both the sides left and right context simultaneously. This enables BERT to capture more language representations, which makes it effective for tasks such as question answering, language inference, and classification. Results displayed by benchmarks like GLUE, MultiNLI, and SQuAD show BERT's success, its ability to downstream tasks with minimal task-specific changes makes it better than previous models. **[4]**

Traditional Computer Vision models rely on supervised learning with labelled data, this limits their ability to generalize to new tasks without additional training, models like ResNet (He et al., 2016) and EfficientNet (Tan and Le, 2019) requires additional labelled data to recognize new visual capabilities. Another emerging alternative is learning visual representation from raw text in images by using the data present over internet. OpenAI's CLIP (Contrastive Language-Image Pre-training) solves the problem by training on a large-scale dataset of 400 million (image, text) pairs, this enables zero-shot transfer to many tasks without task-specific training, tasks such as OCR, action recognition, and fine-grained object classification. This is similar to NLP models like BERT and GPT. CLIP shows potential for scalable, task-specific models in computer vision. **[5]**

Word representation models have showed significant progress in capturing semantic and syntactic relationships. Early approaches like word2vec (Mikolov et al., 2013) used prediction-based models to learn word embedding by using local context windows, these proved to be the best in tasks such as word analogy and similarity. Count-based models, rely on global co-occurrence matrices which provides a broader statical perspective but they suffer from inefficiency.

GloVe (Pennington et al., 2014) bridges these approaches by combining global matix factorisation with local context window learning, this provides statistical efficiency and strong performance on analogy, similarity and named entity recognition tasks. This shows that the distinction between count-bases and prediction-based models may be less fundamental that previously thought. GloVe's success shows that value of capturing global and local information in word representations. **[6]**

SentencePiece is a language-independent subword tokenizer to handle raw text for neural-based text processing tasks such as Neural Machine Translation (NMT). SentencePiece allows subword model training directly from raw sentences which provides fully end-to-end language-agnostic system. This enhances flexibility and efficiency in handling various languages.

By conducting experiments on English-Japanese translation, the model shows comparable performance to traditional tokenizer. SentencePiece's open-source implementation, available in both C++ and Python, ensures reproducibility, making it a valuable tool for multilingual text processing and research. **[7]**

Previous studies such as those comparing character-level CNNs with word-level models, have shown that deeper networks often do not translate to better performance in practical applications, especially when training data is abundant.

The proposed deep pyramid CNN model aims to improve on previous work by balancing the efficiency of shallower networks, which can be just as effective but less complex and therefore perform better, according to the findings of past studies, with depth. Current deep architecture-based models are referenced by the authors, who suggest that more effective design strategies are required to maintain high accuracy on assignment tasks, particularly those requiring the categorization of subjects and classification of sentiment. In addition, it contributes to the ongoing debate regarding the optimization of neural network models for natural language processing by describing a low-complexity model that effectively models long-range dependencies in text. **[8]**

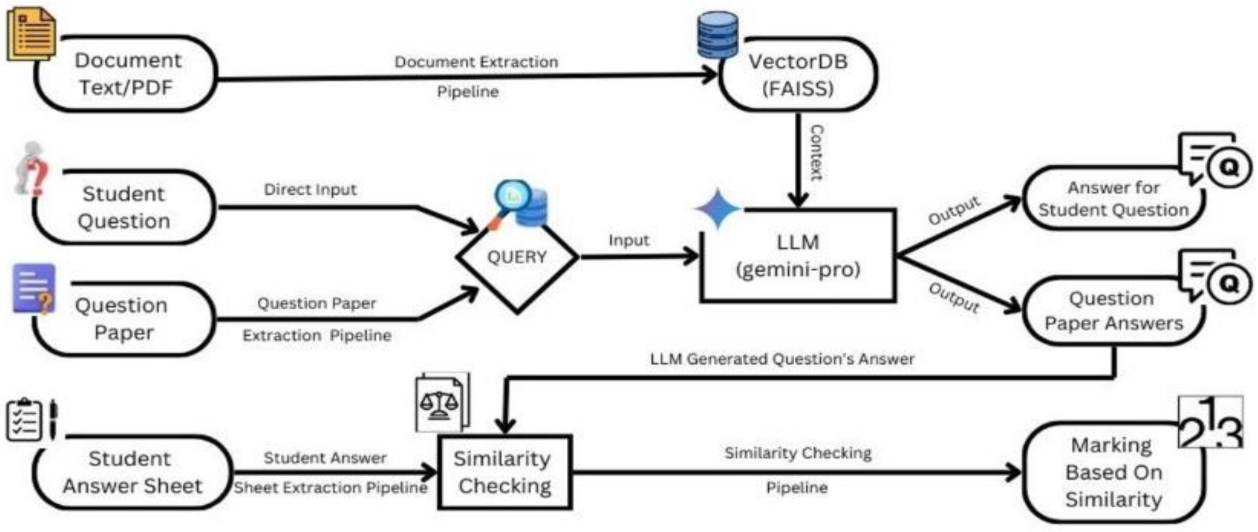
As a potent substitute for conventional statistical machine translation (SMT), neural machine translation (NMT) has drawn interest. NMT models are constructed utilizing a single end-to-end neural network, as opposed to SMT, which depends on distinct components. A limiting issue, particularly for longer phrases, has been found in early models like the encoder-decoder architecture, which encode a source sentence into a fixed-length vector. In their discussions of this bottleneck, Cho et al. (2014b) and Pouget-Abadie et al. (2014) emphasized the need for more adaptable models.

Recent developments have concentrated on attention mechanisms that enable NMT systems to "soft-search" pertinent portions of the source phrase while producing each target word in order to overcome these constraints. Performance has greatly improved since the advent of attention-based models like RNN search. By enabling improved alignment between source and target words and boosting robustness for lengthier phrases, this architecture performs better than previous approaches. Consequently, attention-based NMT models have shown a notable advancement in the area, achieving outcomes that are on par with the most advanced phrase-based systems. But there are still issues to be resolved, such as managing uncommon or unfamiliar terms. **[9]**

Mikolov et al. (2013) presented the continuous Skip-gram model, which is a useful technique for learning distributed word representations that include syntactic and semantic links. Subsampling common words to increase speed and rare word representation, as well as Negative Sampling for effective training, are important advancements. Due to the model's limited ability to handle phrases, techniques for encoding multi-word phrases using single tokens or vector combinations were developed. Analogical reasoning and word similarity identification are two examples of natural language processing tasks that have greatly advanced thanks to extensions that have concentrated on increasing training efficiency and optimizing hyperparameters. **[10]**

APPLIED METHODOLOGY

The proposed system works in two phases: Question Generation and Answer Sheet Evaluation. In the Question Generation phase, the professor uploads a reference document, such as a subject textbook or any other material in PDF format, which serves as the primary data source. The text from the PDF is extracted using the PdfReader library, after which it is divided into smaller, manageable chunks using the RecursiveCharacterTextSplitter. These chunks are transformed into vector representations using Google Generative AI Embeddings and stored in a FAISS (Facebook AI Similarity Search) database. This vectorization process enables the system to search and retrieve relevant information from the textbook. Once the text is processed and embedded, questions and corresponding answers are generated using the Gemini Pro model, allowing the professor to prepare comprehensive and context-relevant exam questions with maximum ease. The diagram below shows how the flow of the project works:


Document Text/PDF
Document Extraction Pipeline
VectorDB (FAISS)
Context
Student Question
Direct Input
QUERY
Input
LLM (gemini-pro)
Output
Answer for Student Question
Output
Question Paper Answers
Question Paper
Question Paper Extraction Pipeline
LLM Generated Question's Answer
Student Answer Sheet
Student Answer Sheet Extraction Pipeline
Similarity Checking
Similarity Checking Pipeline
Marking Based On Similarity

A. Question Generation Pipeline

This pipeline focuses on generating exam questions and their respective answers from the study materials provided as input by the professors. The following steps outline the process:

Upload Reference Textbook (PDF):

The professor uploads a reference document, usually the subject textbook in PDF format. This document will serve as the primary source i.e. as the dataset for generating questions and answers.

Text Extraction using PdfReader:

The text from the uploaded PDF is extracted using the PdfReader library, which processes the document and converts the content into a structured text format, via this step, clean text is fed into the AI model.

Dividing Text into Chunks using RecursiveCharacterTextSplitter:
After the text is extracted, it is divided into smaller chunks using the RecursiveCharacterTextSplitter. This helps reduce the size of the long documents by breaking them down into smaller or more manageable portions of text, ensuring that the model can process the information effectively. Chunking helps in efficient embedding and retrieval.

Convert Text Chunks into Vectors using the GoogleGenerativeAIEmbeddings:
Each chunk of text is then passed to the GoogleGenerativeAIEmbeddings model, which converts the text into vector representations. These vectors capture the semantic meaning of the text, enabling efficient search and question generation later. This step transforms raw text into embeddings, that can be stored in a vector database.

Store Vectors in the FAISS Database:
The vector representations of the text chunks are stored in a FAISS (Facebook AI Similarity Search) database.
FAISS is used because it has the capability to handle large-scale vectors efficiently. The process of searching the vector also becomes easy thus, making it ideal for retrieval of relevant information from the material that is given as the input.

Question and Answer Generation using the Gemini Pro Model:
A conversational chain is created using the Gemini Pro model. When the professor provides an instruction, like for example, "Generate questions on Chapter 1," the model processes the instruction referring to the embedded textbook i.e. the input data, and generates relevant questions and answers. This process leverages the AI's ability to understand the context and content of the textbook for the formation of the questions.
Output: The generated questions and answers are presented to the professor, who can use them or refine them as per their requirements.

B. Answer Sheet Evaluation Pipeline
This pipeline automates the grading process for students. The handwritten answer sheets of the students, i.e. their answers and the AI-generated answers are compared, and based on the similarities, the grades are assigned. The steps involved are as follows:

1. Upload Question Paper (PDF):
The professor uploads the question paper that was given to the students and is now given as the input in PDF format, for generating reference answers for comparison.

2. Extract Questions Sequentially from the Question Paper:
The system processes the uploaded question paper and extracts the questions in a sequential order, storing them in a list format. This step ensures that each question is treated as an independent entity for the comparison of the answers written by the students.

3. Generate AI-based Answers using the Gemini Pro Model:
For each extracted question, the Gemini Pro model is used to generate corresponding answers. These AI-generated answers act as reference answers that will be compared to the student's responses.

4. Upload the Handwritten Answer Sheet:
The professor uploads the scanned image of the student's handwritten answer sheet. This document will be processed to extract the student's responses.

5. Extract Student Answers using OCR:
Optical Character Recognition (OCR) is applied to the uploaded handwritten answer sheet to extract the

student's written answers. The extracted text is stored sequentially, matching the questions from the question paper.

6. Convert AI Answers and Student Answers into Vectors:

Both the AI-generated answers and the student's handwritten answers are converted into vector representations. This allows for a mathematical comparison between the two sets. Each answer is represented as a point in a high-dimensional space, where similar answers will have vectors close to each other.

7. Apply Cosine Similarity for Answer Comparison:

Cosine similarity is used to compare the AI-generated answers and the student's answers. Cosine similarity measures the cosine of the angle between two vectors in the vector space, with a similarity score ranging from 0 (no similarity) to 1 (perfect match). This score indicates how close the student's answer is to the AI-generated reference answer.

8. Advanced Comparison using Keyword Extraction:

In addition to cosine similarity, Natural Language Processing (NLP) techniques are applied to extract keywords from both the AI and the student's answers. These keywords are compared to ensure that the key concepts are addressed in the student’s response. This keyword matching ensures a more nuanced and detailed grading process.

9. Evaluate Answer Coherence using Sentence Transformer:

A sentence transformer model (specifically, "paraphrase-MiniLM-L6-v2") is used to ensure that the student's answer is coherent and meaningful. The transformer checks for sentence-level paraphrasing and semantic similarity. This model assesses the overall structure and flow of the student's response, ensuring that even rephrased or slightly altered answers are considered for marks.

10. Assign Marks Based on Similarity and Keyword Matching:

Based on the cosine similarity score and keyword matching results, the system assigns a mark for each question. If the answer has a high similarity score (close to 1) and contains the necessary keywords, full marks are awarded. The grading is done on a scale from 0 to 1, with thresholds set to determine how many marks are awarded based on the range of similarity and keyword overlap. This ensures that there is consistency in all the grades. This eliminates the problem of unfair grading.

Final Output: The system produces a final score for the student's exam, indicating the marks awarded for each question.

RESULTS

The project was tested using multiple textbooks and student answer sheets from various academic subjects and levels. The automated question-and-answer generation feature provided professors with a flexible tool to quickly generate quizzes, exams, and practice materials, directly from the uploaded textbooks or any kind of materials. The system can create a diverse range of questions, including factual, conceptual, and descriptive questions, aligning with the subject, accurately with reference to the material in the textbooks. This significantly reduced the time and effort required to manually create assessments, enabling more efficient lesson planning and also reducing the stress of the professors.

The system also generated accurate answers to the generated questions, allowing the professors to review both the questions and answers corresponding to them. Apart from this, the students can also use the software to boost their learning process as they can understand what type of questions can be asked and what kind of answers are to be written to those particular questions.

The automated marking system was tested with a variety of answer sheets, comparing AI-generated answers with student responses using cosine similarity and keyword extraction. The cosine similarity method effectively handled both factual and descriptive answers by converting the text into vectors and comparing them, ensuring that even if students used different wording or phrasing, their answers were accurately assessed. The system also used keyword extraction to identify critical terms in descriptive answers, further enhancing the grading accuracy by ensuring all key points were addressed.
The combined approach of cosine similarity and keyword extraction produced reliable grading results, demonstrating high accuracy in both factual and descriptive responses. This automated process reduced the professor's grading workload and provided fair and consistent marks for students, handling both short factual answers and longer descriptive ones with ease. Overall, the system significantly improved the efficiency and reliability of the assessment process in educational environments.

CONCLUSION

This project showcases the potential of integrating generative AI models, vector databases, and natural language processing (NLP) techniques to automate essential educational tasks. The system not only generates questions from textbook content but also grades student responses, offering a scalable and efficient solution for educators. By leveraging cosine similarity, keyword extraction, and AI-generated answers, the system ensures accurate grading, even for descriptive responses. Future improvements will focus on enhancing handwriting recognition accuracy for handwritten answer sheets, making the system more adaptable. Additionally, expanding language support will enable broader adoption in multilingual educational environments, making the solution more inclusive and versatile.

REFRENCES

1. Yuning Du, Chenxia Li, Ruoyu Guo, Xiaoting Yin, Weiwei Liu, Jun Zhou, Yifan Bai, Zilin Yu, Yehua Yang, Qingqing Dang, et al. Pp-ocr: A practical ultra lightweight ocr system. arXiv preprint arXiv:2009.09941, 2020.
2. Hannah Kim and Young-Seob Jeong. 2019. Sentiment classification using convolutional neural networks. Applied Sciences 9, 11 (2019), 2347.
3. Yuchen Li. Synergizing optical character recognition: A comparative analysis and integration of Tesseract, Keras, Paddle, and Azure OCR. University of Sydney, 2023.
4. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
5. Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, and Jack Clark. Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020, 2021.
6. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014.
7. Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 66–71, 2018.
8. Xiang Zhang, Junbo Zhao, and Yann LeCun. Deep pyramid convolutional neural networks for text categorization. arXiv preprint arXiv:1706.02446, 2017.

9. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.
10. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, pages 3111–3119, 2013.