WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

"Automatic Answer Checker System"

Mr. Chetan Nagnath Shinde[1], Prof. Pramod jadhao[2]

[1] Department of MCA ,Trinity Academy Of Engineering , Pune , India,
[2]Assistant Professor of MCA ,Trinity Academy Of Engineering , Pune , India

ABSTRACT :

This research presents a Python-based Automatic Answer Checker System capable of evaluating student-written descriptive answers by comparing them with model answers using natural language processing (NLP) and machine learning. The system automates the time-consuming task of manual checking by analyzing semantic similarity and keyword relevance. This paper highlights the model's capability to provide objective, consistent, and efficient evaluation, especially in educational environments where large-scale assessments are conducted. A Flask-based web interface is used to allow user interaction with the model. The project focuses on text preprocessing, vectorization using TF-IDF, similarity comparison with cosine similarity, and model deployment using Flask..

KEYWORDS : Automatic Answer Checker, NLP, TF-IDF, Cosine Similarity, Flask, Education Technology

INTRODUCTION

With the evolution of educational technologies, automating subjective answer evaluation is gaining prominence. Traditional manual checking is often inconsistent, time-intensive, and prone to human error or bias. An automatic answer checker system aims to mitigate these challenges by evaluating descriptive answers through intelligent algorithms that analyze the semantic meaning and structure of student responses.
This project introduces a system where a model answer is compared to student input using NLP techniques. The similarity is quantified using cosine similarity over TF-IDF vectors. A web interface built using Flask enables seamless integration, allowing teachers or institutions to upload questions, model answers, and receive scored outputs for student responses.

LITREATURE SURVEY/BACKGROUND

Automated essay grading dates back to the 1960s, with systems like Project Essay Grade (PEG) and e-rater. More recently, NLP advancements have enabled more accurate comparison methods.

- **Shermis & Burstein (2013)** demonstrated that statistical and NLP-based automated scoring systems could correlate strongly with human grading.
- **Yadav et al. (2019)** used cosine similarity and semantic similarity techniques for short-answer grading, showing promising results in accuracy and scalability.
- **TF-IDF and Cosine Similarity** remain widely adopted due to their simplicity and interpretability in measuring textual similarity.
- **Flask**, a lightweight web framework in Python, is frequently used in deploying machine learning models with REST APIs for educational tech solutions.

These foundational tools and studies inform the system proposed in this paper.

PROPOSED WORK

1. System Overview:

System Overview
The system automatically compares a student's answer to a predefined model answer and assigns a similarity score. Key objectives include:

- Reducing evaluation time
- Improving fairness and consistency
- Deploying an intuitive and accessible interface

Core Components:

- **Input Interface**: User inputs model and student answers.
- **Preprocessing Module**: Text is cleaned, tokenized, and normalized.
- **Vectorization Module**: Converts text into TF-IDF vectors.
- **Similarity Engine**: Computes cosine similarity and generates a score.
- **Web Interface**: Displays score and allows multiple submissions.

2. System Architecture

1. **User Interface (Frontend)**
    - Takes student and model answers as input
    - Sends data to backend for processing
2. **Text Preprocessing**
    - Lowercasing, stopword removal, lemmatization
    - Tokenizes and normalizes textual data
3. **Feature Extraction**
    - TF-IDF vectorization transforms text into numerical features
4. **Similarity Computation**
    - Cosine similarity determines the closeness of student answer to model answer
    - Output is a percentage-based score
5. **Result Display**
    - Final score is returned to user via Flask web interface
    - Optionally stores logs for further analysis

METHODOLOGY

Data Collection

- Sample question-answer pairs are collected from academic sources.
- Human-marked datasets are used for benchmarking.

Preprocessing

- Conversion to lowercase
- Stopword removal (e.g., “is”, “the”)
- Lemmatization and punctuation removal

Vectorization

- TF-IDF is used to numerically represent both model and student answers.

Scoring

- Cosine similarity is applied on vectors to obtain a similarity score between 0 and 1.
- This score is mapped to a percentage scale to resemble marks.

Web Interface

- Flask is used to build the front-end form and connect it to the scoring engine.

RESULT AND DISCUSSIONS

The system was tested on 50 descriptive answers across various domains such as science and history. Evaluation showed:

- **Accuracy**: Approx. 85% agreement with human scoring
- **Speed**: Instantaneous scoring per submission
- **Consistency**: No subjective bias observed

Short and straightforward answers produced higher confidence in scoring. Limitations were observed in evaluating highly creative or paraphrased answers with indirect language, which could be addressed in future versions using semantic models like BERT.

CONCLUSION AND FUTURE WORK

This paper presents a scalable and effective Automatic Answer Checker System using TF-IDF and cosine similarity. It can substantially aid in automating descriptive answer evaluations in educational settings. With a Flask interface, the system is easy to deploy and use.

Future Enhancements:

- Integrating semantic similarity models (e.g., BERT, SBERT)

- Multilingual support
- Rubric-based scoring
- Feedback generation

REFERENCES

[1] Shermis, M. D., & Burstein, J. (2013). Handbook of Automated Essay Evaluation: Current Applications and New Directions.
[2] Yadav, A., Yadav, S., & Sharma, D. (2019). "A Hybrid Approach for Automatic Short Answer Grading using TF-IDF and LSTM." IEEE.
[3] Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing (3rd ed.)
[4] TF-IDF Documentation: https://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weighting
[5] Flask Documentation: https://flask.palletsprojects.com/
[6] Pedregosa, F., et al. (2011). "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research.
[7] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval.