AI-Powered Answer Evaluation System

Technical Documentation

---

1. Introduction

1.1 Project Title

AI-Powered Answer Evaluation System

1.2 Project Description

The AI-Powered Answer Evaluation System is a web-based application developed using Flask and MySQL that automates the grading of descriptive answers using Natural Language Processing (NLP) and Machine Learning (ML) techniques.

The system evaluates student answers against expected answers using multiple NLP similarity metrics and generates a final score on a scale of 0–10.

The platform supports three user roles:

- Admin
- Teacher
- Student

---

2. System Architecture

2.1 Architecture Overview

The system follows a **Three-Tier Architecture**:

1. Presentation Layer – HTML Templates (Jinja2)
2. Application Layer – Flask (Python)
3. Data Layer – MySQL Database

Architecture Diagram (Logical)

Presentation Layer
↓
Application Layer
↓
Database Layer

---

3. Technology Stack

Backend

- Python 3.8+
- Flask Framework

Database

- MySQL

NLP & Machine Learning Libraries

- NLTK
- Scikit-learn
- Sentence Transformers
- PyTorch (Backend for transformer model)

ML Model Used

SentenceTransformer Model:
paraphrase-MiniLM-L6-v2

This model generates 384-dimensional semantic embeddings for sentences.

---

4. User Roles

4.1 Admin

Responsibilities:

- Add / Update / Delete Students
- Add / Update / Delete Teachers
- View all tests
- View student scores
- Delete student answers

4.2 Teacher

Responsibilities:

- Create tests
- Add questions
- Add expected answers
- View student submissions
- View performance statistics

4.3 Student

Responsibilities:

- Take tests
- Submit answers
- View AI-evaluated scores
- Compare expected answer with their answer

---

5. Database Schema

Tables

1. Admins

- admin_id (Primary Key)
- username
- password

2. Teachers

- teacher_id (Primary Key)
- username
- password

3. Students

- student_id (Primary Key)

- username
- password

4. Tests

- test_id (Primary Key)
- test_name
- teacher_id (Foreign Key)

5. Questions

- question_id (Primary Key)
- question_text
- test_id (Foreign Key)

6. ExpectedAnswers

- answer_id (Primary Key)
- answer_text
- question_id (Foreign Key)

7. StudentAnswers

- answer_id (Primary Key)
- student_id (Foreign Key)
- test_id (Foreign Key)
- question_id (Foreign Key)
- answer_text

- score

---

6. AI Evaluation Algorithm

The evaluation system uses **9 NLP and ML metrics** to calculate answer quality.

Each metric returns a value between 0 and 1.
The final score is scaled to 0–10 using weighted averaging.

---

7. Core NLP Concepts Used

7.1 Tokenization

Tokenization is the process of splitting text into smaller units called tokens (words or sentences).

Example:
"Python is a language" → ["Python", "is", "a", "language"]

Used via NLTK word_tokenize().

7.2 Lemmatization

Lemmatization converts words to their base form.

Example:

- running → run
- studies → study

Used via WordNetLemmatizer (NLTK).

7.3 TF-IDF (Term Frequency – Inverse Document Frequency)

TF-IDF is a numerical statistic that measures how important a word is in a document compared to a collection of documents.

Term Frequency (TF)

TF measures how frequently a word appears in a document.

TF = (Number of times term appears in document) / (Total words in document)

Inverse Document Frequency (IDF)

IDF measures how unique or rare a word is across multiple documents.

IDF = log(Total number of documents / Number of documents containing the word)

Rare words get higher IDF score.

Why TF-IDF is Important?

Common words like "is", "the", "and" get lower scores.
Important words like "programming", "algorithm" get higher scores.

7.4 Cosine Similarity

Cosine Similarity measures similarity between two vectors by calculating the cosine of the angle between them.

Formula:

Cosine Similarity = $(A \cdot B) / (||A|| \times ||B||)$

Where:

- A and B are vector representations of sentences

- "·" is dot product
- ||A|| is magnitude of vector A

Range:

- 1 → Identical
- 0 → Completely different
- -1 → Opposite

In this project:

- Sentences are converted to TF-IDF vectors
- Cosine similarity is calculated

7.5 Sentence Embeddings

Sentence embeddings convert sentences into fixed-length numeric vectors.

Used Model:
SentenceTransformer ('paraphrase-MiniLM-L6-v2')

This captures semantic meaning beyond word matching.

Example:
"Python is a programming language"
"Python is used for coding"

These may have high semantic similarity even if words differ.

7.6 Naive Bayes Classifier

Multinomial Naive Bayes is a probabilistic machine learning algorithm based on Bayes' Theorem.

Bayes' Theorem:

$P(A|B) = (P(B|A) \times P(A)) / P(B)$

Assumption:
Features are independent (naive assumption).

In this system:

- Text is converted to count vectors
- Naive Bayes predicts probability that student answer matches expected answer
- Probability score contributes to final grade

7.7 Sentiment Analysis (VADER)

Sentiment analysis determines whether a sentence is positive, negative, or neutral.

Used library:
NLTK SentimentIntensityAnalyzer (VADER)

Returns:

- Positive score
- Negative score
- Neutral score
- Compound score (used here)

7.8 Coherence Score

Measures similarity in length between expected and student answer.

Formula:

Coherence = min(word_count1, word_count2) / max(word_count1, word_count2)

Closer to 1 → Similar length

7.9 Relevance Score

Measures token overlap between expected answer and student answer.

Formula:

Relevance = (Common tokens) / (Total tokens in expected answer)

---

8. Evaluation Metrics and Weights

Metric	Weight
Exact Match	15%
Partial Match	10%
Cosine Similarity	10%
Sentiment Analysis	5%
Enhanced Sentence Match	10%
Multinomial Naive Bayes	10%
Semantic Similarity	10%
Coherence	10%
Relevance	10%

Total Weight = 90%

Final Score = Weighted Sum × 10

---

9. Evaluation Flow

1. Student submits answer
2. System fetches expected answer
3. Text preprocessing:
    - Tokenization
    - Lemmatization
    - Lowercase conversion
4. Calculate all 9 metrics
5. Apply weights
6. Scale to 0–10
7. Store score in database
8. Display result

---

10. Performance Considerations

- SentenceTransformer model size: ~90MB
- PyTorch backend: ~750MB
- First evaluation may take 30–60 seconds
- Subsequent evaluations are faster due to caching

---

11. Security Considerations

Current Issues:

- Plain text passwords
- Hardcoded secret key
- No CSRF protection

Recommended Improvements:

- Password hashing using werkzeug.security
- Environment variables for secrets
- Flask-WTF for CSRF
- Session timeout implementation

---

12. Future Enhancements

- Grammar checking integration
- Plagiarism detection
- Advanced analytics dashboard
- Time-limited tests
- Multiple choice support
- PDF report generation

---

13. Conclusion

The AI-Powered Answer Evaluation System provides automated grading of descriptive answers using advanced NLP and Machine Learning techniques.

By combining:

- TF-IDF
- Cosine Similarity
- Sentence Embeddings
- Naive Bayes
- Token Matching
- Sentiment Analysis

The system ensures a balanced and accurate scoring mechanism that evaluates both syntactic and semantic similarity.