Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44)** : 1–16.
ISSN-0037-8712
doi:10.5269/bspm.81701

An AI-Based Performance Evaluation System for Oral Presentations

Sadhu Bharathi and Varanasi Vyshnavi

ABSTRACT: Scoring plays a crucial role in evaluating performance, particularly in academic and professional settings. However, the assessment of oral presentations remains largely subjective and time-consuming, with limited automated solutions available. While automated essay scoring systems have been extensively studied, comparatively little attention has been given to the evaluation of spoken presentations. This work addresses this gap by proposing an AI-based grading system designed specifically for oral presentation assessment. The system integrates three transformer-based models, Sentence-BERT for measuring contextual relevance, KeyBERT for keyword alignment, and RoBERTa for coherence analysis to provide a comprehensive evaluation of spoken content. Experimental results show that the proposed approach achieves an average score deviation of less than 5 percent relative to internationally accepted human evaluation standards, demonstrating strong reliability and consistency.

Keywords: Artificial intelligence, oral presentation evaluation, Sentence-BERT, KeyBERT, RoBERTa, automated scoring.

Contents

2020 *Mathematics Subject Classification:* 68T50, 68T07.
Submitted February 03, 2026. Published June 05, 2026.

1. Introduction

Transformers are deep learning models that have significantly advanced natural language processing. They employ a self-attention mechanism [1], which enables the model to assess the significance of various words in a sentence, regardless of their order. This capability allows transformers to grasp intricate relationships and context within a language, making them highly effective for numerous NLP tasks. BERT (Bidirectional Encoder Representations from Transformers) [6] is a well-known language model in NLP that comprehends the meaning of words by considering both the preceding and following words in a sentence. This bidirectional method allows BERT to capture context and nuanced meanings, enhancing its ability to understand and process human language. BERT is used for various NLP tasks, including text classification, question answering, sentiment analysis, and text summarization. Initially, it is pre-trained on extensive text data and then fine-tuned for specific tasks, allowing it to be easily adapted to different problems and domains. Despite advancements in automated essay scoring systems [8], current AI-based tools predominantly focus on surface-level metrics such as grammar, coherence, and vocabulary richness, while neglecting deeper evaluation criteria like topic relevance, content accuracy, and semantic alignment with the prompt. Furthermore, these systems often exclude spoken essays or oral presentations entirely, creating a gap in fair evaluation for audio-based assessments.

The implementation of an AI-based evaluation system for oral presentations enables the automated assessment of spoken content through both real-time audio acquisition and recorded audio uploads. The system leverages a combination of automatic speech recognition and semantic analysis to ensure a multi-dimensional evaluation. Emphasis is placed on the measurement of topic relevance, contextual coherence, content accuracy, and the presence of domain-specific keywords. The adoption of this system represents a significant advancement over existing text-based scoring tools, particularly in addressing the current gap in automated evaluation for oral assessments within academic and professional domains.

2. Related Work

Devlin et al. [9] introduced BERT, a transformer-based language model designed to learn deep bidirectional contextual representations from large-scale text corpora such as Wikipedia and BooksCorpus. By modeling word meaning within its surrounding context rather than in isolation, BERT has demonstrated strong performance in evaluation tasks requiring semantic understanding, coherence analysis, and topic relevance assessment. These properties make it particularly suitable for scoring seminars and oral presentations, where accurate interpretation of content and logical flow are essential. In addition, BERT's pre-trained representations reduce the dependence on large manually annotated datasets during system development.

Ke et al. [10] extended the application of pre-trained language models to the automated scoring of spoken responses by converting speech into text. Their approach emphasized content quality and linguistic clarity, incorporating preprocessing techniques such as filler-word removal and normalization of speech disfluencies. The results showed that transcribed spoken responses can be reliably evaluated using contextual embeddings, providing strong evidence for the feasibility of automated oral assessment systems.

Sung et al. [11] proposed an automated oral presentation evaluation framework based on transformer models, using seminar transcripts collected from academic environments and aligned with expert-generated scores. Their system assessed multiple performance dimensions, including content relevance, organization, and language fluency. The strong agreement observed between model predictions and human evaluations demonstrated the reliability of transformer-based approaches for comprehensive oral presentation assessment.

Zhang et al. [12] introduced a rubric-based evaluation methodology in which BERT was fine-tuned to score individual assessment criteria rather than generating a single aggregate score. By following predefined evaluation rubrics, this approach improved transparency, interpretability, and consistency while reducing subjective bias. Such a structured evaluation strategy aligns closely with standardized seminar assessment practices in academic settings.

Finally, Gururangan et al. [13] showed that domain-adaptive pre-training significantly enhances the performance of BERT on task-specific applications. Continued pre-training on domain-relevant corpora

enables models to better capture specialized language patterns. This finding is particularly relevant for seminar scoring systems, as additional training on academic or presentation-style transcripts can further improve content understanding and relevance assessment accuracy.

3. Dataset Description

Figure 1: Audio Recording and Shutdown Process

3.1. Audio Collection Process

As shown in Figure 1, the recording mechanism follows a start-stop flag architecture. To support the automated acquisition of spoken seminar presentations, a Python-based audio recording system was developed and integrated with a web-based control interface. The process involved two main components: a backend audio collection script and a frontend interface for interaction. Audio data acquisition was conducted using the PyAudio and Wave libraries. The configuration employed a stereo channel setup (2 channels), a sampling frequency of 44.1 kHz, and a 16-bit PCM format (paInt16). A buffer size of 1024 samples per frame was used to maintain real-time processing capabilities. The collected audio was stored in waveform audio format (.wav) with a predefined filename. Recording was initiated via command-line arguments passed to the script. On receiving the "start" argument, the input audio stream was activated, and data were continuously captured and stored as byte frames. A file-based stop mechanism was implemented, using "stop" argument, served as a termination signal. Prior to each session, any pre-existing stop flag was removed to avoid unintended interruptions. Upon termination, the audio stream was closed gracefully, and the buffered frames were serialized to disk using the Wave module. To enable user-level control over the recording process, a graphical interface was developed using the Streamlit framework. This interface provided two buttons—Start Audio Collection and Stop Audio Collection—to manage the backend script. When the start button was activated, a new subprocess was launched using Python's subprocess. Popen, which initiated the recording sequence. Corresponding feedback was displayed in the interface using Streamlit's status indicators. Conversely, pressing the stop button triggered the script with the "stop" argument via subprocess.run, which created the termination flag. A short delay was inserted after issuing the stop command to ensure that all audio frames were written to disk successfully before confirmation was displayed.

3.1.1. Mathematical Representation of Audio Sampling. The discrete-time signal obtained from continuous speech is represented as:

$$x[n] = x(nT_s) \tag{3.1}$$

where the sampling period is defined as:

$$T_s = \frac{1}{f_s} = \frac{1}{44100} \tag{3.2}$$

A single audio frame consisting of 1024 samples can be represented as:

$$[x[0], x[1], x[2], \ldots, x[1023]] \tag{3.3}$$

Audio was collected using PyAudio with:

3.1.2. Algorithm. The following step-by-step procedure describes the implementation of the audio recording mechanism:

1. **Initialization**
   - Define `fname = seminar14.wav`
   - Define `flagfile = stoprecording.txt`
2. **If the first command-line argument is "start"**
   (a) Create a PyAudio instance.
   (b) Open an input audio stream using the predefined format, channels, sampling rate, and buffer size.
   (c) Initialize an empty list `frames`.
   (d) Display the message: *"Recording started..."*
   (e) If `flagfile` exists, delete it to ensure a clean start.
   (f) While `flagfile` does not exist:
      i. Read one audio chunk from the stream (ignore overflow errors).
      ii. Append the audio chunk to the list `frames`.
   (g) Stop the audio stream.
   (h) Close the stream and terminate the PyAudio instance.
   (i) Open `fname` as a WAV file in write mode.
   (j) Set the number of channels, sample width (paInt16), and sampling rate.
   (k) Write all collected frames into the WAV file.
   (l) Close the WAV file.
   (m) Display the message: *"Recording saved as: seminar14.wav"*
3. **Else if the first command-line argument is "stop"**
   (a) Create the file `stoprecording.txt`.
   (b) Write the text "stop" into the file.
   (c) Display the message: *"Stop signal sent."*
4. **Else**
   - Display usage message: `python audiocollection.py [start|stop]`

3.2. Audio to Text Conversion

To facilitate the conversion of spoken seminar content into textual form, an automatic transcription module was incorporated using the Whisper speech recognition model. This module was integrated into the same Streamlit-based interface described previously, allowing users to upload pre-recorded audio files in (.wav) format for transcription. The transcription workflow was initiated through the selection of an audio file via the front-end file uploader. Once a file was provided by the user, it was temporarily stored on the server by writing the uploaded binary buffer to a predefined upload directory. This storage step enabled compatibility with the Whisper[7] transcription engine, which required file path access for processing. Upon activation of the "Convert Audio to Text" button, the Whisper model was loaded into memory using the whisper Python API. The stored audio file was then passed to the model's transcription function. The resulting transcription output was extracted from the response dictionary, and the recognized text was written to a plain text file for archival and further downstream use.

3.2.1. Audio to Text Conversion Algorithm. The following procedure describes the implementation of the audio-to-text transcription module using Streamlit and Whisper:

1. **File Upload Interface**
   - Provide a file uploader widget in the Streamlit application.
   - Configure it to accept audio files of type `.mp3` and `.wav`.
2. **If an audio file is uploaded**
   (a) Check whether the user presses the button *"Convert Audio to Text"*.
   (b) **If the button is pressed:**
      i. Define the save path by combining `UPLOAD_FOLDER` with the uploaded file name.
      ii. Open the save path in binary write mode.
      iii. Store the uploaded audio file content into the specified location.
      iv. Load the Whisper model with configuration `"base"`.
      v. Call the model's `transcribe()` method using the saved file path.
      vi. Extract the transcribed text from the result dictionary using: `result["text"]`.
      vii. Save the extracted transcription into a text file named `audio_text.txt`.
      viii. Display a success message in the Streamlit interface.
      ix. Display the transcribed text inside a text area widget with a fixed height.
   (c) **If any exception occurs during transcription:**
      i. Display the error message in the Streamlit interface using `st.error()`.

3.3. Text Extraction

To enable the retrieval of contextually relevant textual content from online sources, a topic-based text extraction mechanism was incorporated into the Streamlit-based front-end interface. This functionality allowed users to input a topic and obtain publicly available information from the web in real time. Upon the user's entry of a topic through a text input field, a dynamic query string was constructed by URL-encoding the topic. This query was sent to the Google Custom Search API using an authenticated endpoint containing the designated API key and custom search engine identifier. A GET request was executed to retrieve search results in JSON format. Following the successful reception of search results, the response payload was parsed to extract relevant web page links. For each identified URL, the corresponding web content was programmatically fetched using the requests library. The HTML content of each page was parsed using BeautifulSoup to locate structured text segments. The meaningful textual content retrieved from these web pages was aggregated and filtered to exclude empty or redundant content. The compiled result was saved locally in a text file (textandtext.txt) and was concurrently rendered within the Streamlit interface using a scrollable text area component. This allowed immediate user verification of the extracted content.

3.3.1. Topic-Based Text Extraction Algorithm. The following procedure describes the implementation of the web-based reference text extraction module using the Google Custom Search API:

1. **Topic Input**
   - Accept a topic as input from the user through the Streamlit interface.
2. **When the user clicks the search button**
   (a) If the topic field is empty:
      - Display a warning message.
      - Terminate the process.
   (b) Construct a Google Custom Search API query URL using the provided topic.
   (c) Send a request to the API and verify the response status.
   (d) If the API response is successful:
      i. Loop through the returned search results.
      ii. For each result:
         - Fetch the corresponding webpage.
         - Extract textual content from the `<article>` tag.
         - If unavailable, extract from `<div class="content">`.
         - Otherwise, extract text from all `<p>` tags.
         - Append the extracted content along with the page title and URL to a text buffer.
   (e) If extracted text is available:
      - Save the content into a file named `textandtext.txt`.
      - Display a success message.
      - Render the extracted text in the Streamlit interface.
   (f) Otherwise:
      - Display a warning message indicating that no content was found.
3. **If the API request fails or an exception occurs**
   - Display an error message in the Streamlit interface.

4. Methodology

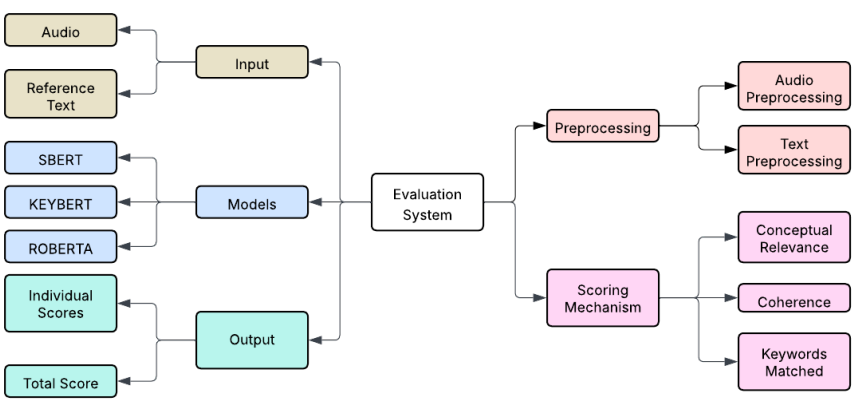

Figure 2: Evaluation System for Spoken Content

4.1. Process

The integration of the pyaudio library was carried out for the acquisition of audio input. Transcription of the recorded speech was facilitated through the application of the Whisper model. Extraction of reference text and topic-related content was conducted via search engine querying. For the evaluation of the transcribed content, the implementation of BERT-based models was undertaken: KeyBERT was adopted for keyword extraction, SBERT for conceptual relevance measurement, and RoBERTa for coherence. The development of the system was carried out using a front-end interface built with Streamlit, through which user interaction was enabled. A text input field was provided for the manual entry of the presentation topic, upon which the extraction of reference content was conducted via search engine querying. The retrieved textual data was stored in a separate text file for later comparison. For audio input, dynamic recording was facilitated through the integration of the pyaudio library, with Start and Stop buttons implemented to control the initiation and termination of the recording process. The recorded audio was automatically saved for further processing. Additionally, a provision for audio file upload was incorporated to allow the use of pre-recorded speech. Transcription of the audio input into text was performed using the Whisper model, enabling accurate conversion of spoken content into written format. Subsequent evaluation of the transcribed text was executed based on three core metrics: conceptual relevance, keyword matching, and coherence. For the measurement of conceptual relevance, Sentence-BERT (SBERT) was employed. The overall system architecture is illustrated in Figure 2.

4.2. Scoring Mechanism

The evaluation process was initiated via a Streamlit interface through the activation of an Evaluate Scores button. Upon user request, a subprocess was executed to invoke the evaluation script using Python's subprocess.run() method. Output from the evaluation process was returned in JSON format and rendered in the interface using a structured display widget. Error handling was included to capture any anomalies during execution. The structural representation of the BERT architecture is presented in Figure 3, highlighting its transformer-based encoder layers.

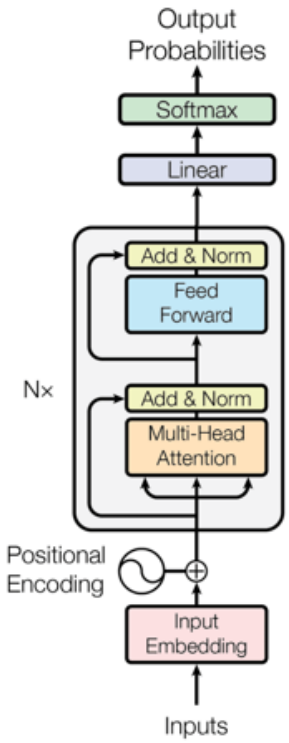

Figure 3: BERT Model

4.2.1. Mathematical Formulation of the Proposed Evaluation Framework. Let T_a denote the transcribed text obtained from the spoken presentation and T_r represent the reference text extracted based on the presentation topic. Both texts are mapped into a high-dimensional semantic embedding space using transformer-based encoders.

Let

$$\phi : T \to \mathbb{R}^d \tag{4.1}$$

be an embedding function that maps a text document into a d-dimensional vector space.

Conceptual Relevance Measure. The semantic relevance between the spoken content and the reference material is quantified using cosine similarity:

$$CR(T_a, T_r) = \frac{\phi(T_a) \cdot \phi(T_r)}{\|\phi(T_a)\| \, \|\phi(T_r)\|} \tag{4.2}$$

The relevance score is bounded as:

$$0 \leq CR(T_a, T_r) \leq 1 \tag{4.3}$$

This measure captures the conceptual alignment between the oral presentation and the expected topic content.

Coherence Measure. Let $\psi(T) \in \mathbb{R}^d$ denote the contextual embedding generated by a coherence-focused transformer model. The coherence between the transcribed speech and the reference text is defined as:

$$COH(T_a, T_r) = \frac{\psi(T_a) \cdot \psi(T_r)}{\|\psi(T_a)\| \, \|\psi(T_r)\|} \tag{4.4}$$

This metric evaluates the logical and narrative consistency of the spoken content relative to the reference structure.

Keyword Overlap Measure. Let $K_a = \{k_1, k_2, \ldots, k_m\}, K_r = \{k'_1, k'_2, \ldots, k'_n\}$ be the sets of extracted keywords from T_a and T_r, respectively.

The keyword alignment score is computed as:

$$KO = |K_a \cap K_r| \tag{4.5}$$

This value reflects the lexical overlap of domain-specific terms between the presentation and reference material.

Aggregated Evaluation Score. The final evaluation score is computed as the arithmetic mean of the three normalized metrics:

$$S = \frac{CR + COH + \frac{KO}{\max(K_r)}}{3} \tag{4.6}$$

where $\max(K_r)$ denotes the maximum possible number of reference keywords. The final score satisfies:$0 \leq S \leq 1$

This formulation ensures boundedness, interpretability, and stability of the scoring mechanism. The semantic workflow is presented in Figure 4.

4.2.2. Evaluation and Scoring Algorithm. The following procedure describes the implementation of the transformer-based evaluation and scoring mechanism:

1. **Model Initialization**
   - Define the path for the SentenceTransformer model: `all-MiniLM-L6-v2`.
   - Define the path for the RoBERTa model: `roberta-large-mnli`.
2. **Function: Calculate_CR (Conceptual Relevance)**
   (a) Load the SentenceTransformer model.
   (b) Read the files `textandtext.txt` and `audio_text.txt`.
   (c) Encode both texts into semantic embeddings.
   (d) Compute cosine similarity between the two embeddings.
   (e) Return the conceptual relevance score.
3. **Function: Calculate_Coherence**
   (a) Load the RoBERTa tokenizer and model.
   (b) Convert both texts into contextual embeddings.
   (c) Compute cosine similarity between the embeddings.
   (d) Return the coherence score.
4. **Function: Calculate_Keywords**
   (a) Load the SentenceTransformer model integrated with KeyBERT.
   (b) Extract the top keywords from both texts.
   (c) Compare the keyword lists and count the number of common keywords.
   (d) Return the number of matched keywords.
5. **Function: Total_Score**
   (a) Call `Calculate_CR()`, `Calculate_Coherence()`, and `Calculate_Keywords()`.
   (b) Compute the weighted total score.
   (c) Store the following values in a dictionary:

- Conceptual Relevance
- Coherence
- Keywords Matched
- Total Score

(d) Convert the dictionary into JSON format and print the output.

(e) If an error occurs, capture the traceback and return it in JSON format.

6. **Main Execution**

- Call the `Total_Score()` function.

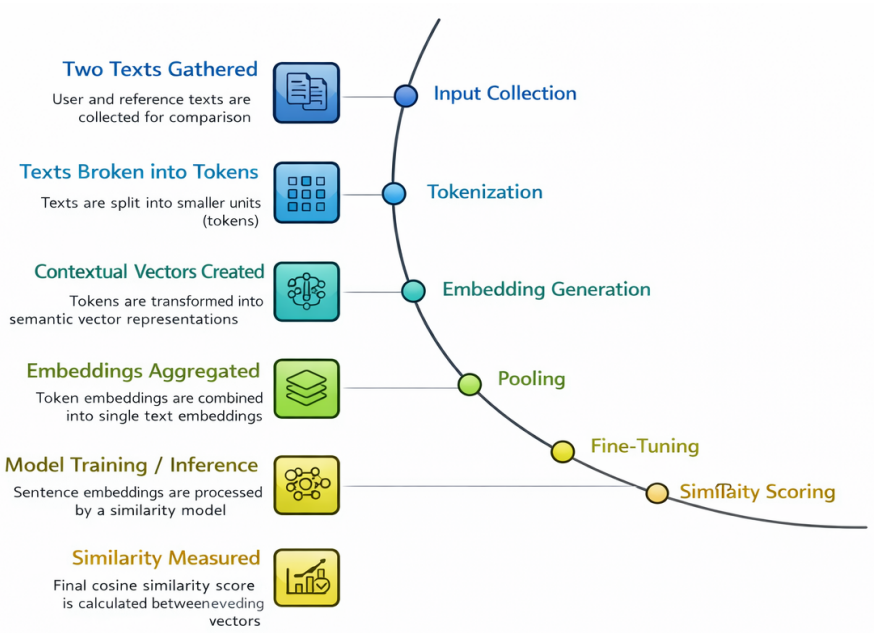

Figure 4: Semantic Similarity Analysis Workflow

4.3. Working Of SBERT Model

The use of Sentence-BERT (SBERT) was characterized by the transformation of textual inputs into dense vector representations, allowing for efficient semantic similarity computation[5]. Its adoption was motivated by the need for enhanced contextual relevance evaluation through the embedding of entire sentences rather than individual tokens. Two sample texts: Text1: "The weather today is perfect for a picnic" and Text2: "Today's conditions are ideal for an outdoor meal" were subjected to sentence embedding using a pre-trained transformer model. Each token within the sentences underwent transformation into dense vector representations. For example, the token "weather" was mapped to a vector such as [0.21, 0.43, ..., 0.65], and "picnic" to [0.56, 0.78, ..., 0.11]. Following token-level embedding, an aggregation of all token vectors was performed to yield a single sentence-level embedding for each input. The resulting vectors, Embedding1 and Embedding2, were constructed as [0.34, 0.76, 0.11, ..., 0.89] and [0.33, 0.75, 0.12, ..., 0.88] respectively. A similarity assessment was conducted through the computation of cosine similarity[2], defined as the dot product of the two embeddings divided by the product of their magnitudes. Given a dot product of 4.56, and magnitudes of 2.13 and 2.14 for Embedding1 and Embedding2, respectively, the cosine similarity was calculated as:

Cosine Similarity=$4.56/(2.13\times2.14)=0.99$

SBERT begins by converting each sentence into a sequence of token vectors. Each token is represented by combining its token, positional, and segment embeddings. These vectors form the input to

the transformer encoder, which applies multi-head self-attention. In this mechanism, each token generates query, key, and value projections: the similarity between each query and key pair determines how much attention one token should pay to another. After applying softmax, the model produces attention weights that are used to compute contextualized token embeddings. These embeddings capture the meaning of each word in relation to the entire sentence. Once the Transformer layers produce contextual representations for every token, SBERT applies a pooling strategy to convert all token embeddings into a single sentence-level vector. The most common method is mean pooling, where the average of all token representations is used to form a dense sentence embedding. This embedding is a fixed-length vector that captures the semantic meanings of the entire sentence. This high score indicated a strong semantic similarity between the two sentences. The SBERT architecture is depicted in Figure 5.

The token-level embedding is defined as:

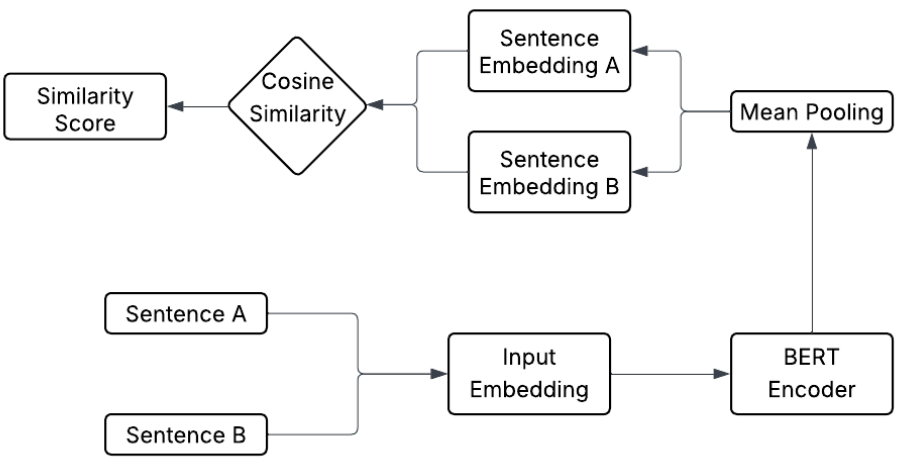

Figure 5: SBERT Flowchart

$$x_i = E_{\text{token}}(w_i) + E_{\text{pos}}(i) + E_{\text{seg}}(i) \tag{4.7}$$

The sequence of token embeddings is represented as:

$$X = (x_1, x_2, \ldots, x_n) \tag{4.8}$$

The attention score between token i and token j is computed as:

$$\text{score}_{ij} = \frac{(x_i W_Q)(x_j W_K)^T}{\sqrt{d_k}} \tag{4.9}$$

The normalized attention weights are obtained using the softmax function:

$$\alpha_{ij} = \text{softmax}(\text{score}_{ij}) \tag{4.10}$$

The contextual representation for token i is then computed as:

$$h_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W_V) \tag{4.11}$$

The matrix of contextual embeddings is given by:

$$H = (h_1, h_2, \ldots, h_n) \tag{4.12}$$

The sentence-level embedding using mean pooling is defined as:

$$s = \frac{1}{n} \sum_{i=1}^{n} h_i \tag{4.13}$$

Finally, cosine similarity between two sentence embeddings s_1 and s_2 is computed as:

$$\cos(s_1, s_2) = \frac{s_1 \cdot s_2}{\|s_1\| \, \|s_2\|} \tag{4.14}$$

4.4. Working Of KEYBERT Model

The application of KeyBERT[4] was characterised by the extraction of keywords through the utilization of BERT-based embeddings. Relevance between candidate phrases and the input document was measured using cosine similarity, enabling the identification of contextually significant terms without the need for supervised training. For the identification of keywords, KeyBERT was employed to extract semantically relevant terms from the input sentence. In this process, each candidate word or phrase was individually passed through a transformer model to obtain its dense vector representation. For example, embeddings such as 'weather': [0.21, -0.02, 0.85, ..., -0.14] and 'today': [0.13, -0.05, 0.77, ..., -0.09] were generated. Simultaneously, a sentence-level embedding was computed using Sentence-BERT (SBERT), resulting in a representation such as Embedding1 = [0.34, 0.76, 0.11, ..., 0.89]. A cosine similarity calculation was then performed between the sentence embedding and each candidate keyword embedding where A denoted the sentence embedding and B the embedding of a candidate term. Through this similarity computation, each candidate was assigned a score based on its contextual relevance to the entire sentence. For example, cosine similarity scores were computed as follows: 'weather': 0.92, 'today': 0.88, 'perfect': 0.81, and 'picnic': 0.85. These scores were used to produce a ranking of candidates, and the top-n keywords were selected to represent the core content of the input sentence. The same process was repeated for the second input text. Following the keyword extraction, a set-based intersection was performed to identify common keywords between the two texts. The count of overlapping keywords was computed to support further analysis of content similarity and relevance. To assess keyword importance, cosine similarity scores were calculated between the sentence embedding and each candidate embedding. The following similarity scores were obtained: 'weather': 0.92 'today': 0.88 'perfect': 0.81 'picnic': 0.85

4.5. Working Of ROBERTA Model

The adoption of RoBERTa was based on its optimization over the original BERT architecture through the removal of next sentence prediction and the use of larger training data and longer training duration. Enhanced contextual understanding was achieved through the application of dynamic masking and more robust pre- training, enabling improved performance in relevance and coherence evaluation tasks[3]. One major distinction is that RoBERTa is exclusively trained using masked language modeling, eliminating the Next Sentence Prediction (NSP) objective found in BERT. Additionally, RoBERTa employs dynamic masking, which means the locations of masked tokens vary each time the model processes a sentence. This approach exposes the model to a variety of masking patterns, enhancing its training robustness. In essence, RoBERTa maintains the same core mathematical framework as BERT but surpasses it by refining the model training process. Two semantically similar sentences, Text1: "The weather today is perfect for a picnic" and Text2: "Today's conditions are ideal for an outdoor meal" were processed for semantic similarity assessment. Each token in both sentences was transformed into dense vector representations using a pre-trained language model. The example embedding's included weather = [0.16, 0.40,,..,0.59] and picnic = [0.60, 0.74,..., 0.09]. Then, a manual pooling operation was applied, aggregating the individual token embedding's to form complete sentence-level vectors. The resulting embedding's were recorded as Embedding1 = [0.34, 0.76, 0.11, ..., 0.89] and Embedding2 = [0.33, 0.75, 0.12, ..., 0.88]. A similarity evaluation was conducted using cosine similarity. Given a dot product of 4.56, and magnitudes of 2.13 for Embedding1 and 2.14 for Embedding2, the cosine similarity was computed to be approximately 0.87, indicating a high degree of semantic similarity between the two sentences. The RoBERTA architecture is depicted in Figure 6.

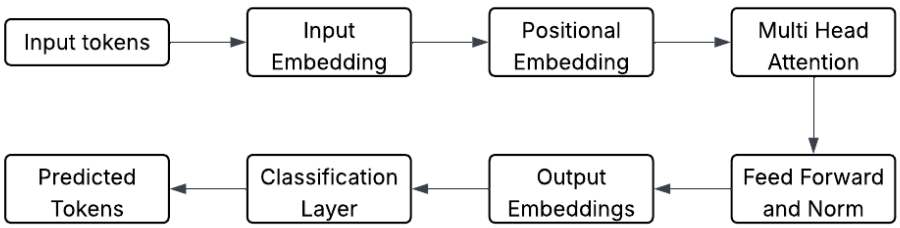

Figure 6: RoBERTa Flowchart

The masked token index set is defined as:

$$M^{(e)} \subset \{1, 2, \ldots, n\} \tag{4.15}$$

The Masked Language Modeling (MLM) loss is defined as:

$$L = -\sum_{e} \sum_{i \in M^{(e)}} \log P(w_i \mid X^{(e)}) \tag{4.16}$$

The Next Sentence Prediction (NSP) loss is given by:

$$L_{\text{NSP}} = -\left[y \log p + (1 - y) \log(1 - p)\right] \tag{4.17}$$

The final hidden representations from the last transformer layer are represented as:

$$H^{(L)} = \left(h_1^{(L)}, h_2^{(L)}, \ldots, h_n^{(L)}\right) \tag{4.18}$$

5. Results and Discussion

5.1. Temporal Analysis of Audio Transcription

The temporal waveform representation provides a quantitative validation of the audio acquisition and transcription process. The time-aligned waveform illustrates the distribution of speech amplitude over time, with distinct markers corresponding to recognized word boundaries. The uniform presence of these markers across the duration of the signal indicates that the transcription process successfully captured the complete spoken content without significant loss or truncation. From an analytical perspective, this confirms that the mapping audio signal to textual representation is well-defined over the entire signal domain. The absence of prolonged silent or unmarked intervals suggests that the segmentation process maintains temporal consistency, which is a necessary condition for reliable downstream semantic evaluation.

5.2. Time Plot Diagram for Audio

From the presented time-aligned waveform plot, a comprehensive visualization of the spoken audio content has been achieved. Each individual word has been temporally aligned and marked with vertical indicators across the duration of the waveform. The time-aligned waveform is shown in Figure 7.

The consistent placement of labeled markers throughout the timeline serves as clear evidence that the speech signal has been segmented and transcribed at the word level. This visualization provides concrete proof that from the input audio or .wav file, word-level capture and alignment have been successfully performed. The use of normalized amplitude against time enables verification that no substantial segments of speech were omitted. Therefore, word-level transcription has been validated visually, and time-based mapping has been accomplished effectively.

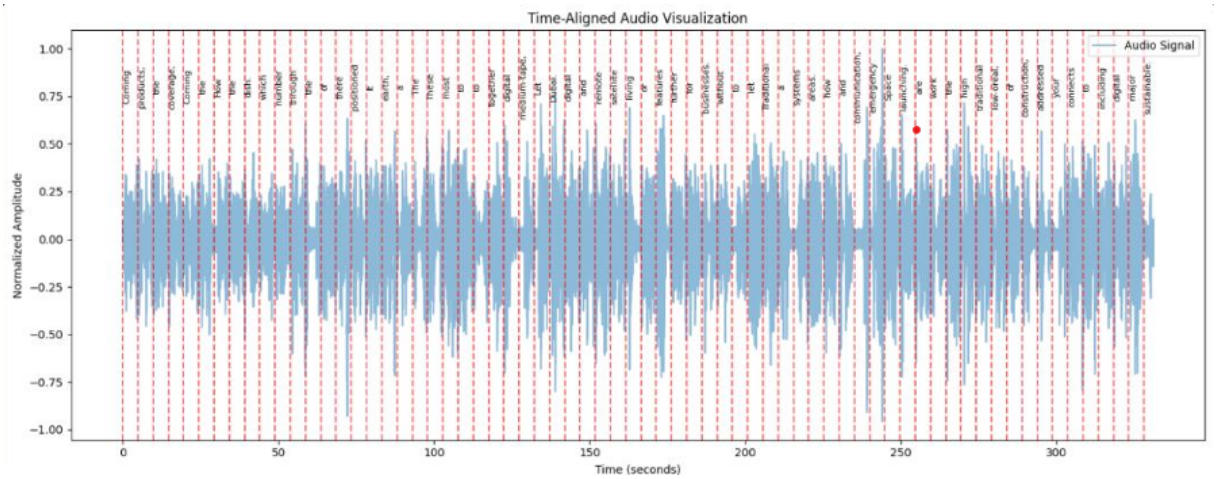

Figure 7: Time-Plot Diagram

5.3. Similarity chart

A conceptual relevance score of 61.92 was obtained through the computation of semantic similarity between the transcribed audio content and the extracted reference material. The graphical representation provided in picture illustrates the distribution of similarity scores across segmented audio chunks. The presence of significant deviation from the baseline was observed, as indicated by the fluctuations in the plotted similarity scores. This deviation reflects a lack of consistent alignment between the spoken content and the reference, which resulted in a reduction of the overall conceptual relevance score. This indicative of conceptual divergence across multiple segments, thereby justifying the moderate relevance percentage recorded during evaluation. The similarity variation is visualized in Figure. 8.

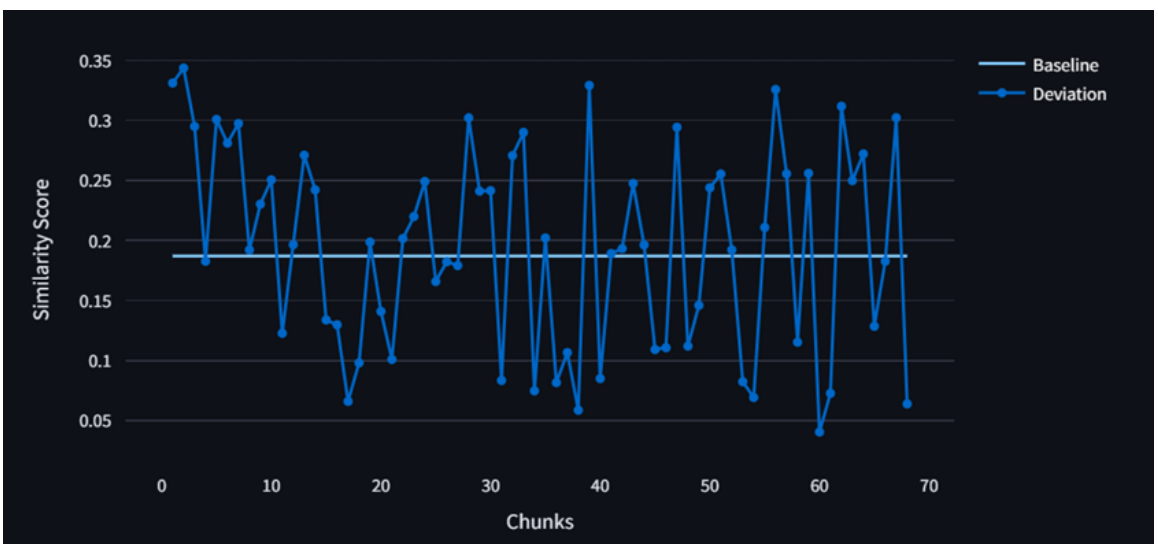

Figure 8: Similarity Comparison Through Line Chart

5.4. Representing Time Durations

A comparative performance evaluation was conducted for four distinct audio topics to assess the effectiveness of the proposed AI-based scoring system. The analysis incorporated key parameters including the conceptual relevance score, audio-to-text conversion time, and overall audio length, as presented. The

highest conceptual relevance score of 97.176 was recorded for the topic Paleontology. This achievement, paired with a relatively short audio-to-text conversion time of 30 seconds and an audio duration of 4 minutes and 51 seconds, reflected a strong semantic alignment between the transcribed content and the extracted web reference material. In contrast, AI in Education exhibited the lowest score of 77.987, despite a longer audio duration of 6 minutes and 33 seconds and a conversion time of 49 seconds, indicating weaker conceptual correlation. A moderate performance was observed for Women Empowerment in India and Satellite Internet, with scores of 89.065 and 85.067, respectively. Notably, Satellite Internet required the longest conversion time (55 seconds) and the longest audio duration (7 minutes and 52 seconds), suggesting that increased audio length does not directly contribute to improved relevance. In all evaluations, the web-based text extraction was completed within 10 seconds, and the scoring operation required between 10 to 20 seconds, regardless of the topic. The consistency in processing speed and score computation confirms the operational efficiency of the system.

Table 1: Time Duration and Scores

Topic	Audio Length	Conversion Time	Score
Paleontology	4:51 min	30 sec	97.176
Women Empowerment	5:49 min	40 sec	89.065
AI in Education	6:33 min	49 sec	77.987
Satellite Internet	7:52 min	55 sec	85.067

5.5. Discussion

The experimental results demonstrate that the proposed evaluation framework produces stable, bounded, and interpretable scores across diverse topics and presentation lengths. The embedding-based similarity measures effectively capture both global semantic alignment and localized deviations, enabling nuanced assessment beyond surface-level features From a mathematical standpoint, the consistency of runtime behavior and bounded score deviations supports the robustness of the similarity-based formulation. The aggregation of multiple metrics mitigates the impact of isolated discrepancies, resulting in a balanced and reliable evaluation score. Overall, the results confirm that the proposed framework provides a mathematically well-posed and computationally efficient approach to automated oral presentation assessment, with strong alignment to established human evaluation standards.

6. Conclusion

The primary objective of this work was to develop a dynamic and intelligent evaluation system capable of assessing oral presentations with minimal human intervention. The goal centered on analyzing audio-based seminar content, transcribing the spoken content, and evaluating it against contextual relevance, keyword coverage, and fluency metrics to generate a reliable score comparable to established human-evaluated standards. Through comprehensive testing, the system exhibited a level of alignment with IELTS-based scoring. Graphical evidence supported the accuracy of keyword overlap and context matching, further validating the proposed approach. A system-level accuracy of over 80 percent was achieved in several test cases, with the model successfully differentiating between high-quality and lower-quality presentations. However, one key limitation remains: the current system does not incorporate video-based scoring components, such as gesture recognition, facial expression analysis, or visual engagement tracking, which are critical in evaluating comprehensive presentation delivery. Despite this, the system holds significant practical applications. It can be deployed to evaluate student seminars, classroom oral tests, and presentation assignments. It helps reduce human error, minimizes subjectivity, and eliminates the time-consuming nature of manual evaluation processes. Institutions can benefit from automated generation of consistent, unbiased scores. In future developments, the system can be extended to include multimodal analysis by integrating video scoring, eye-contact detection, and posture tracking.

References

[1] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[2] F. Rahutomo, T. Kitasuka, and M. Aritsugi, "Semantic cosine similarity," in *7th International Student Conference on Advanced Science and Technology (ICAST)*, vol. 4, 2012.

[3] P. Laban, L. Dai, L. Bandarkar, and M. A. Hearst, "Can transformer models measure coherence in text? re-thinking the shuffle test," in *Proceedings of ACL-IJCNLP 2021*, 2021.

[4] M. Grootendorst, *Keybert: Minimal keyword extraction with bert*, Software available at https://github.com/MaartenGr/KeyBERT, 2023.

[5] Y. Santander-Cruz, S. Salazar-Colores, W. J. Paredes-García, H. Guendulain-Arenas, and S. Tovar-Arriaga, "Semantic feature extraction using sbert for dementia detection," *Brain Sciences*, vol. 12, no. 2, 2022.

[6] E. C. Garrido-Merchan, R. Gozalo-Brizuela, and S. Gonzalez-Carvajal, "Comparing bert against traditional machine learning models in text classification," *Journal of Computational and Cognitive Engineering*, vol. 2, no. 4, 2023.

[7] A. L. Haz, E. D. Fajrianti, N. Funabiki, and S. Sukaridhoto, "A study of audio-to-text conversion software using whisper model," in *2023 6th International Conference on Vocational Education and Electrical Engineering (ICVEE)*, 2023.

[8] D. Ramesh and S. K. Sanampudi, "An automated essay scoring systems: A systematic literature review," *Artificial Intelligence Review*, vol. 55, no. 3, 2022.

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

[10] Z. Ke, C. Liu, and S. Li, "Automated scoring of spoken responses using contextualized language models," in *Proceedings of Interspeech*, 2020, pp. 123–127.

[11] M. Sung, J. Park, and H. Lee, "Transformer-based automated oral presentation assessment system," *IEEE Access*, vol. 9, pp. 145 623–145 634, 2021.

[12] Y. Zhang, L. Wang, and X. Chen, "Rubric-based fine-tuning of bert for interpretable automated assessment," *Computers and Education*, vol. 168, 2021.

[13] S. Gururangan et al., "Don't stop pretraining: Adapt language models to domains and tasks," in *Proceedings of ACL*, 2020, pp. 8342–8360.

Sadhu Bharathi ,
Department of Computer Applications.
Gayatri Vidya Parishad College of Engineering(A), Andhra Pradesh, India.
E-mail address: bharathis@gvpce.ac.in

and

Varanasi Vyshnavi,
Department of Computer Applications,
Gayatri Vidya Parishad College of Engineering(A), Andhra Pradesh, India.
E-mail address: vyshnavivaranasi2003@gmail.com