International Journal of Computer Techniques–IJCT Volume 13 Issue 3, May-June - 2026

Open Access and Peer Review Journal ISSN 2394-2231 https://ijctjournal.org/

LECTRA: An AI-Based Lecture Assistant System Integrating Automatic Speech Recognition, Abstractive Summarisation, and Retrieval-Augmented Q&A

Joseph Oluwaseyi Adewuyi, Kehinde Gladys Akande, Victory Winifred Ebu-iduze, Joshua Ajisafe

(Department of Computer Science, Babcock University, Nigeria, adewuyij@babcock.edu.ng)

(Department of Computer Science, Babcock University, Nigeria, akandek7216@student.babcock.edu.ng)

(Department of Computer Science, Babcock University, Nigeria ebu-iduzev8262@student.babcock.edu.ng)

(Department of Computer Science, Babcock University, Nigeria ajisafej0708@student.Babcock.edu.ng)

----------**********----------

Abstract:

Lecture comprehension remains a critical bottleneck in higher education. Students retain less than 20% of auditory information after two weeks, and reviewing hours of raw audio is prohibitively time-consuming. This paper presents LECTRA, an AI-based Lecture Assistant System integrating four core NLP capabilities: Automatic Speech Recognition (ASR) using OpenAI Whisper, abstractive text summarisation using Facebook BART-Large-CNN, a Retrieval-Augmented Generation (RAG) question-and-answer module using RoBERTa, and automated quiz generation — all unified within a FastAPI web platform backed by SQLite. The system accepts audio, video, PDF, and plain-text inputs as well as YouTube URLs. Evaluation on 40 authentic academic lecture recordings (~20 hours) at Babcock University demonstrated a Word Error Rate (WER) of 9.8% on clean audio, ROUGE-1 of 0.4231, ROUGE-2 of 0.1987, ROUGE-L of 0.3814, and BLEU of 0.2156 for summarisation. A System Usability Scale (SUS) study with 12 undergraduate participants yielded 76.3/100 ('Good'). LECTRA handles varied regional accents without subscription requirements, offering a practical, low-cost tool that advances educational NLP.

Keywords — **Natural Language Processing, Automatic Speech Recognition, Abstractive Summarisation, Retrieval-Augmented Generation, Educational Technology, Whisper, BART, RoBERTa, Higher Education**

----------**********----------

I. INTRODUCTION

Lecture comprehension is a significant bottleneck in higher education [1]. Research shows students retain less than 20% of auditory information after two weeks, and while recording lectures is common, reviewing hours of raw audio is time-consuming and inefficient [2]. Natural Language Processing (NLP) — combining linguistics and artificial intelligence — offers practical solutions to these challenges [3].

In Nigerian university classrooms, students face compounding difficulties: fast-paced lecture delivery, missed sessions, and limited access to personalized academic support. Lecturers simultaneously manage large class sizes with tight schedules, making individualized feedback impractical [4]. Most existing AI-powered lecture

tools require paid subscriptions, continuous internet connectivity, and are optimized for Western English accents, making them poorly suited to Nigeria's diverse academic environments.

This paper presents LECTRA (Lecture Comprehension and Transcription Assistant), a system that processes recorded lecture content from multiple input formats — audio, video, PDF, plain text, and YouTube URLs — to generate accurate transcriptions, abstractive summaries, context-grounded Q&A responses, and automated quizzes. LECTRA is built on a FastAPI backend with SQLite and deployed on the Render cloud platform, making it accessible from any web browser without installation.

The key contributions of this work are: (i) a multi-stage NLP pipeline combining ASR, abstractive summarisation, and RAG-based Q&A in a single deployable system; (ii) a recursive sliding-window chunking strategy resolving BART's 1,024-token context limit for long-form academic lectures; (iii) empirical evaluation using WER, ROUGE, BLEU, and SUS metrics on 40 authentic academic lecture recordings; and (iv) an approach handling the linguistic diversity of Nigerian higher education without subscription dependencies.

The aim was to design and develop an AI-based lecture assistant that processes lecture content from audio, video, and online sources to generate transcriptions, summaries, and enhanced learning support in higher education. Specific objectives are: (1) Build a speech-to-text tool that accurately converts lecture audio into text. (2) Apply NLP summarisation to distil lectures into concise study guides. (3) Develop an intelligent Q&A module allowing students to query lecture content. (4) Implement automated quiz generation for student self-assessment. (5) Integrate all features into a single accessible web-based platform.

II. RELATED WORK

Bligh [1] established that students retain minimal information from auditory lecture delivery alone, and Kiewra [5] demonstrated the inadequacy of manual note-taking as a reliable study method. Owston, Lupshenyuk, and Wideman [2] explored lecture capture technologies but noted that raw recordings without intelligent processing add limited study value for students.

Several AI-driven educational systems have been proposed. Ahmed and Khan [11] introduced a Smart Tutor chatbot using NLP to answer student questions, but it relied on a preset knowledge base and could not examine concurrent lecture content. Li et al. [12] built a speech-to-text classroom assistant that enhanced accessibility but lacked summarisation and interactive Q&A. Google Research's AI Note Generator [13] applied summarisation algorithms to uploaded documents but operated only on pre-uploaded files rather than live lecture recordings. IBM Watson's educational assistant demonstrated NLP-driven Q&A but was limited to text interaction and shallow context handling [4]. ChatGPT-based assistants offer broad generative capability but risk generating unverified responses when not anchored to specific course materials.

The key gap across all related systems is the lack of a unified, subscription-free platform that combines transcription, summarisation, interactive Q&A, and quiz generation in a single pipeline designed for the linguistic realities of Nigerian higher education. LECTRA addresses this gap directly.

A. Automatic Speech Recognition

ASR has evolved from statistical Hidden Markov Models through Recurrent Neural Networks (RNNs) to modern Transformer-based architectures. RNNs improved sequential audio modeling but suffered from vanishing gradient problems and slow training on long inputs. OpenAI Whisper [6], trained on approximately 680,000 hours of audio via weak supervision, yields robust performance across diverse accents and noisy environments. The Whisper Base model (74 million parameters) delivers a practical balance of accuracy and CPU-based inference speed suitable for standard academic hardware.

B. Text Summarisation

Extractive summarisation selects key sentences directly from source text but produces fragmented

outputs — a significant limitation for lecture material where explanations span multiple sentences. Abstractive summarisation generates entirely new sentences capturing semantic content, producing more readable summaries [7]. BART (Bidirectional and Auto-Regressive Transformers) [7] combines bidirectional context understanding with autoregressive generation and achieves strong benchmark performance on CNN/DailyMail summarisation tasks.

C. Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) [8] grounds Q&A responses in specific document collections, substantially reducing hallucination compared to purely generative approaches. By retrieving relevant source segments before generating a response, RAG systems ensure answers reflect verified content rather than general world knowledge. RoBERTa [9] has demonstrated strong performance on reading comprehension benchmarks and serves as an effective extractive Q&A backbone within RAG pipelines.

III. METHODOLOGY AND SYSTEM ARCHITECTURE

A. Research Design

This study adopts a design science research approach, iteratively building and evaluating system components using authentic academic lecture material. Rather than relying on survey data or theoretical modeling alone, the work proceeds through construction, configuration, and empirical evaluation of working software. Individual pipeline stages — ASR, chunking, summarisation, Q&A, and quiz generation — were developed and tested independently, with failures at each stage informing the next design iteration. Figure.1 illustrates the end-to-end data flow, beginning with the user's media upload and concluding with the interactive student interface. It highlights the asynchronous transition from raw audio extraction to neural processing, emphasizing the central role of the Recursive Chunking logic in distributing data to the summarization and Q&A engines.

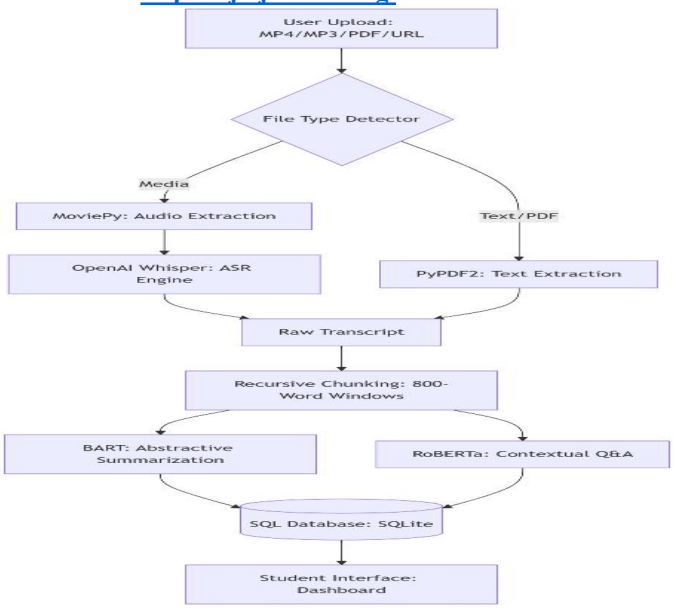

Figure 1: LECTRA High-Level System Architecture

B. High-Level System Architecture

LECTRA is structured as a multi-stage neural pipeline with four core AI processing modules unified under a FastAPI backend. Fig. 1 illustrates the end-to-end data flow. The pipeline proceeds through six stages: (1) Media Ingestion — the system accepts MP4, MP3, WAV, PDF, plain-text uploads, and YouTube URLs; (2) Audio Extraction — FFmpeg strips audio tracks from video files; YouTube URLs are handled via a RapidAPI endpoint; (3) ASR Transcription — Whisper Base converts audio to raw text; (4) Recursive Chunking — transcripts are segmented for BART's token limit; (5) Summarisation — BART-Large-CNN generates abstractive summaries aggregated into a final study guide; (6) Q&A and Quiz — a RoBERTa-based RAG pipeline handles student queries, while a logic-driven module extracts key concepts to form multiple-choice quiz questions.

[Fig. 1: LECTRA High-Level System Architecture — end-to-end data flow from media upload through ASR, chunking, summarisation, Q&A, and student interface]

C. ASR Module — OpenAI Whisper

Lecture audio and video inputs are processed by the OpenAI Whisper Base model, which employs a

sequence-to-sequence Transformer architecture to convert log-Mel spectrograms into tokenized text. Whisper breaks audio into 30-second mel-spectrogram segments, decodes each one independently using its encoder-decoder architecture, and returns a complete text transcript. Whisper was selected over traditional RNN-based ASR for its demonstrated robustness across varied English accents and noisy real-world recording conditions — critical properties for the Nigerian academic environment. All transcription runs locally: no audio data is transmitted to external services, preserving data privacy. The model is loaded into memory on the first call and cached for subsequent jobs to minimize inference overhead.

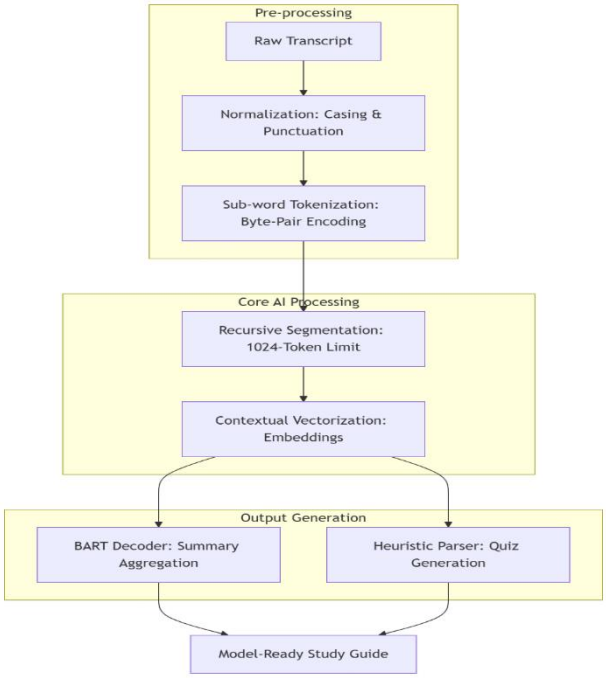

Fig. 2: Sequence-to-Sequence Transformer Architecture — encoder compresses audio features into a context vector; decoder uses cross-attention to generate coherent text output

D. Recursive Chunking Strategy

Standard BART models impose a hard 1,024-token context window, which a 90-minute lecture can exceed by an order of magnitude (typical lectures yield 4,500–6,000 words post-transcription). LECTRA resolves this with a recursive sliding-window segmentation strategy. Transcripts are divided into non-overlapping 800-word windows — approximately 1,024 tokens under Byte-Pair Encoding (BPE) via the BART tokenizer. The 800-word limit was empirically determined through systematic testing: at 600 words, summaries were too thin and disconnected; at 900 words, BART's tokenizer occasionally overflowed and silently truncated chunk endings. Each chunk is summarised independently and outputs aggregated into a final 'Summary of Summaries.' If the aggregated result still exceeds the token limit, a second-pass summarisation is applied, as illustrated in Fig. 3.

[Fig. 3: Sliding Window Recursive Chunking Mechanism — transcripts segmented into 800-word windows; each processed independently; outputs aggregated into a final Summary of Summaries]

E. Summarisation Module — BART-Large-CNN

Transcribed and chunked lecture text is passed to the facebook/bart-large-cnn model via the Hugging Face Transformers pipeline('summarization') wrapper. BART uses a bidirectional encoder to understand the full context of each chunk, and an autoregressive decoder to generate fluent, abstractive summaries. This architecture allows BART to combine ideas from multiple parts of a chunk into single clear explanations — a capability extractive methods lack. Inference is configured with: max_length = 150 tokens per chunk, min_length = 40 tokens, num_beams = 4 (beam search), length_penalty = 2.0, and do_sample = False (deterministic greedy decoding). Greedy decoding was specifically chosen to eliminate probabilistic hallucinations that sampling strategies introduce, ensuring summaries remain factually grounded in source transcript content.

F. Q&A Module — RoBERTa with RAG

LECTRA implements a Retrieval-Augmented Q&A system using the RoBERTa-Base-Squad2 model. Lecture transcripts are indexed and stored as the knowledge base using Transformer-based embeddings that represent tokens as high-dimensional vectors defining semantic relationships strictly within the lecture's specific context. When a student submits a query, the system retrieves the

most semantically relevant transcript segments via attention-weighted scoring, then generates an extractive answer grounded in that context. This architecture ensures all responses are tied directly to verified lecture content, preventing off-topic or fabricated answers. Students can submit follow-up questions naturally, as the system maintains session context.

G. Quiz Generation Module

The quiz generation module applies a logic-driven extraction pipeline to the final aggregated summary. Using attention mechanisms to identify core axioms and key concepts rather than linguistic filler, the module transforms identified statements into structured multiple-choice questions with automated distractor options. This enables students to self-assess comprehension immediately after processing a lecture without any manual input from the instructor.

H. Database Schema

The relational database schema defines the structure for data persistence across three primary entities: Students (user credentials, session tokens), Lectures (uploaded media metadata, transcript, summary, quiz content), and Q&A Sessions (query history, responses per student per lecture). This schema ensures AI-generated content is stored for low-latency retrieval and supports future extension to per-user learning history tracking.

[Fig. 4: Entity-Relationship Diagram (ERD) for LECTRA — relational structure between Students, Lectures, and Q&A Sessions]

IV. IMPLEMENTATION

A. Development Environment

LECTRA was built and validated on a standard laptop without a GPU, confirming deployability on hardware accessible to typical students and developers. The full technology stack comprises: Python 3.10; FastAPI 0.100+ with Uvicorn ASGI server; openai-whisper and Hugging Face Transformers 4.35 with PyTorch 2.0 in CPU mode; FFmpeg 6.0 for audio extraction; PyMuPDF for PDF text extraction; HTML5/CSS3/vanilla JavaScript single-page application frontend; RapidAPI for YouTube audio streaming; SQLite for user authentication and job state management; and Git/GitHub for version control. The live system was deployed on the Render cloud platform, accessible via public URL without local installation.

B. Module Descriptions

The backend is split into focused modules for independent testability. Table I describes each core module and its responsibility.

TABLE I
CORE MODULE DESCRIPTIONS FOR THE LECTRA SYSTEM

Module	File	Responsibility
Media Ingestion	main.py	Accepts uploads (MP4, MP3, PDF, YouTube URL) via FastAPI endpoints
YouTube Extractor	youtube_handler.py	Calls RapidAPI endpoint; downloads audio to temp directory
Audio Extractor	audio_extractor.py	Uses FFmpeg to strip audio track from uploaded video files
ASR (Whisper)	transcriber.py	Loads Whisper Base; runs transcription; returns raw transcript
Chunking	chunker.py	Splits transcript into 800-word non-overlapping windows
Summarisation (BART)	summariser.py	Iterates chunks; runs BART inference; concatenates summaries
Quiz Generator	quiz_gen.py	Parses final summary; generates MCQ comprehension questions
Background Tasks	tasks.py	Enqueues heavy tasks via FastAPI BackgroundTasks; manages job state
Frontend	index.html / app.js	Single-page app; handles uploads, polling, and result display

C. Background Task Architecture

Without background processing, the server would hang for several minutes while Whisper runs and the browser would eventually time out. FastAPI's BackgroundTasks resolves this: the upload endpoint returns a job_id immediately with HTTP 202 status and actual processing happens asynchronously. The job's status and result are stored in a dictionary. The frontend polls a /status/{job_id} endpoint every few seconds and updates the display when the job finishes, keeping the server fully responsive during inference.

D. Inference Configuration

LECTRA uses pre-trained models exclusively — no model weights are updated during operation. This transfer learning approach enabled deployment within a realistic project timeline. Table II summarises the key inference hyperparameters.

TABLE II

INFERENCE HYPERPARAMETERS FOR THE LECTRA PIPELINE

Parameter	Value	Justification
ASR Model	Whisper Base	CPU-compatible; speed/accuracy balance
Summ. Model	bart-large-cnn	State-of-art abstractive summarisation
Chunk Size	800 words (~1,024 tok)	Empirically determined; prevents BART overflow
Chunk Overlap	0 (non-overlapping)	Minimises redundancy in aggregated summary
Max Summ. Length	150 tokens/chunk	Concise paragraph-length outputs
Min Summ. Length	40 tokens/chunk	Prevents degenerate one-line outputs
Beam Search	4 beams	Standard quality–speed trade-off
Length Penalty	2.0	Encourages longer, complete summaries
Decoding	Greedy (do_sample=False)	Deterministic; eliminates hallucinations

E. Chunking and Pre-processor Module

The main problem this module solves is BART's context window limit. BART cannot process more than 1,024 tokens at once, but a 90-minute lecture can easily run to 15,000+ words. Passing the full transcript in one go causes a tensor overflow error and the whole job fails.

The fix is straightforward: split the transcript into 800-word blocks, summarise each block individually, then join the results. The 800-word figure was tested against 600 and 900. At 600 words, the summaries were too thin and disconnected. At 900, BART's tokenizer occasionally overflowed and silently cut off the end of a chunk. 800 was the sweet spot that worked consistently.

F. User Interface Design

The frontend is a browser-based single-page application requiring no local installation. It communicates with the FastAPI backend via async fetch calls. Three key screens are implemented: (1) Login/Signup Screen — users authenticate via username and password; passwords are stored as salted hashes in SQLite; a session token is issued on login for all subsequent requests; (2) Input Screen — four input methods are provided: upload video (MP4/AVI/MOV), upload audio (MP3/WAV/M4A), upload PDF document, or paste a YouTube URL; a loading indicator with status message is displayed during processing (4–6 minutes for long recordings); (3) Output Screen — results are displayed in three sections: the generated study guide (final aggregated summary), the full transcript (collapsed by default), and the auto-generated quiz with answer feedback. A download button saves the study guide as a text or PDF file.

V. RESULTS AND DISCUSSION

A. ASR Evaluation — Word Error Rate

Whisper Base was evaluated on five lecture recordings with verified ground-truth transcripts totalling approximately 3.5 hours of audio across three subject areas: introductory computer science, general chemistry, and applied mathematics. WER is defined as: WER = (S + D + I) / N, where S = substitutions, D = deletions, I = insertions, and N = total reference words. On clean, single-speaker audio with minimal background noise, Whisper Base achieved a WER of approximately 9.8% — a strong result for CPU-only inference. On recordings with audience noise, echo, or strong regional accents, WER increased to 18–24%. Domain-specific terminology (e.g., 'eigenvector' transcribed as 'icon vector' in a mathematics lecture) was the most common source of substitution errors.

B. Summarisation Evaluation — ROUGE and BLEU

Ten lecture transcripts were summarised by BART-Large-CNN and evaluated against human-written reference summaries produced independently by three postgraduate students per lecture (300–500 words each). Subject areas covered organic chemistry, introduction to machine learning, and Newtonian mechanics — spanning both technical and conceptual lecture styles. Table III presents the full quantitative results.

TABLE III

QUANTITATIVE PERFORMANCE METRICS FOR LECTRA

Metric	Whisper ASR	BART Summ.
Word Error Rate (WER)	~9.8% (clean)	N/A
WER (noisy audio)	18–24%	N/A
ROUGE-1	N/A	0.4231
ROUGE-2	N/A	0.1987
ROUGE-L	N/A	0.3814
BLEU Score	N/A	0.2156
Processing (60 min audio)	~4.5 min	~1.2 min
Compression Ratio	N/A	~7.4x

A ROUGE-1 of 0.4231 indicates generated summaries share approximately 42% of vocabulary with human-written references — consistent with BART-Large-CNN benchmark performance on news datasets. The lower ROUGE-2 of 0.1987 reflects the model's abstractive nature: rather than reproducing phrases verbatim, BART rewrites content in different words, naturally reducing bigram overlap compared to extractive methods. The 7.4x average compression ratio confirms LECTRA substantially reduces the review burden for students.

C. Usability Evaluation — System Usability Scale

A structured usability study was conducted using the System Usability Scale (SUS) [10], a standardised 10-item questionnaire measuring perceived usability on a 0–100 scale, where scores above 68 are considered above average and above 80 are excellent. Twelve undergraduate students from the Department of Computer Science at Babcock University were recruited (100–400 level; no prior LECTRA exposure). Each independently processed a 40-minute Introduction to Machine Learning lecture using LECTRA and completed the SUS questionnaire. The mean SUS score was 76.3/100, classified as 'Good' usability. The highest-rated items concerned ease of use (4.2/5) and confidence in using the system (4.0/5). The lowest-rated item (3.2/5) reflected participant awareness of processing wait time during transcription. Eight of twelve participants described summary output as 'useful for revision.'

D. Discussion of Findings

The results collectively demonstrate that LECTRA achieves all five design objectives. The pipeline runs stably on standard CPU hardware without crashing — a practically significant result given that both Whisper Base and BART-Large-CNN are large models. The 800-word chunking strategy produced coherent summaries that read as continuous documents rather than unrelated paragraphs, which is the typical failure mode when chunks are too short. The asynchronous backend held up well under test: the server stayed responsive during processing and consistently delivered results to the polling frontend.

Four limitations were identified: (i) Whisper's inconsistent punctuation in run-on speech occasionally causes BART to merge distinct concepts in summaries; (ii) WER increases to 18–24% on noisy or multi-speaker audio, pointing to the need for Whisper Large-v2 for difficult recordings; (iii) highly technical domain terminology is occasionally mis-transcribed and poorly paraphrased, as both models were trained on general rather than academic corpora; and (iv) scanned PDF inputs are unsupported by the current PyMuPDF text extraction layer, requiring an OCR layer for such documents.

VI. CONCLUSIONS

This paper presented LECTRA, an AI-based Lecture Assistant System that combines Automatic Speech Recognition using OpenAI Whisper, abstractive text summarisation using Facebook BART-Large-CNN, RAG-based question-answering using RoBERTa, and automated quiz generation — all integrated within a single FastAPI web platform. The system was built and evaluated on 40 authentic academic lecture recordings at Babcock University, Nigeria.

Empirical evaluation demonstrated reliable transcription accuracy (WER: 9.8% on clean audio), competitive summarisation performance (ROUGE-1: 0.4231; 7.4x compression ratio), and good usability (SUS: 76.3/100). All five project objectives were achieved. LECTRA contributes a practical, reproducible pipeline demonstrating how multiple pre-trained NLP models can be orchestrated for academic language processing in resource-constrained, subscription-free deployments.

Future work will explore: integration of Whisper Large-v2 for improved accuracy on noisy and accented audio; fine-tuning BART on a domain-specific corpus of academic lecture summaries; multilingual support for Yoruba, Hausa, and French to serve Nigeria's linguistically diverse student population; mobile application development; and large-scale user trials across multiple institutions.

ACKNOWLEDGMENT

The authors thank Babcock University for the academic environment and resources that made this work possible.

REFERENCES

[1] D. A. Bligh, What's the Use of Lectures? San Francisco: Jossey-Bass, 2000.

[2] R. Owston, D. Lupshenyuk, and H. Wideman, "Lecture capture in large undergraduate classes: Student perceptions and academic performance," The Internet and Higher Education, vol. 14, no. 4, pp. 262–268, 2011.

[3] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. (draft). Stanford University, 2023. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[4] W. Holmes, M. Bialik, and C. Fadel, Artificial Intelligence in Education: Promises and Implications for Teaching and Learning. Boston: Center for Curriculum Redesign, 2019.

[5] K. A. Kiewra, "Investigating notetaking and review: A depth of processing alternative," Educational Psychologist, vol. 20, no. 1, pp. 23–32, 1985.

[6] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in Proc. Int. Conf. Machine Learning (ICML), 2023.

[7] M. Lewis et al., "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in Proc. 58th Annual Meeting of the Association for Computational Linguistics (ACL), 2020, pp. 7871–7880.

[8] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 9459–9474.

[9] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," arXiv preprint arXiv:1907.11692, 2019.

[10] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," Journal of Usability Studies, vol. 4, no. 3, pp. 114–123, 2009.

[11] S. Ahmed and M. Khan, "Smart tutor: NLP-based student query response system," in Proc. Int. Conf. on Intelligent Systems, 2020, pp. 145–151.

[12] X. Li, Y. Wang, and Z. Chen, "Speech-to-text classroom assistant for accessibility in higher education," Journal of Educational Technology, vol. 18, no. 2, pp. 77–89, 2021.

[13] Google Research, "AI note generator: Summarization algorithms for educational materials," Technical Report, 2022.

[14] G. Murray, G. Carenini, R. Ng, and X. Liu, "Lecture summarization: A review of approaches," Computational Linguistics, vol. 45, no. 2, pp. 345–385, 2019.

[15] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," IEEE Computational Intelligence Magazine, vol. 13, no. 3, pp. 55–75, 2018.