International Journal of Advanced Scientific Research and Engineering Trends

ISSN: 2456-0774

Volume 10 Issue 1s (2026) | Pages: 234-245

Speech-To-Text Summarization System For Smart Note-Taking

Shubham Pandey[1], Prince Raj[2], Rahul Dutta[3], Eliyas Mulla[4], Vidya Medhe[5]

[1,2,3,4,5]Department of Computer Engineering, Bharati Vidyapeeth College of Engineering, Pune, Maharashtra, India

Peer Review Information

Type: *Article*

Received: *02 April 2026*

Revised: *28 April 2026*

Accepted: *05 June 2026*

Published: *23 June 2026*

Abstract

The rapid convergence of Automatic Speech Recognition (ASR) and Natural Language Processing (NLP) has fundamentally transformed the digital accessibility landscape, enabling the development of sophisticated applications capable of transcribing and summarizing spoken content in real-time. This comprehensive research report investigates the architectural, linguistic, and computational frameworks essential for constructing a highly accurate Speech-to-Text and Text-to-Summarization application, with a specialized focus on multilingual and low-resource Indian languages such as Hindi and Marathi.[1] While traditional acoustic models have demonstrated remarkable proficiency in high-resource contexts, adapting these systems to linguistically diverse and morphologically complex environments presents profound technical challenges. Through a rigorous examination of state-of-the-art acoustic models—including OpenAI's Whisper, Wav2Vec 2.0, and Conformer—this study evaluates their feature encoding mechanisms, quantization processes, and adaptability to code-mixed speech.[3] Concurrently, the report analyses advanced abstractive summarization engines like BART, T5, and PEGASUS, emphasizing their capacity to generate coherent, factually consistent summaries from conversational transcripts.[1] Special attention is devoted to the AI4Bharat ecosystem, specifically the Indic BART model, which leverages Devanagari script unification to overcome data scarcity in Indian language processing.[8] The study further explores end-to-end system architecture, highlighting sophisticated evaluation metrics such as Word Error Rate (WER) and BERTScore.[10] Finally, the report addresses the socio-technical implications of such applications, including their impact on educational cognitive load, user experience design paradigms, and the critical ethical considerations surrounding voice-inferred data privacy and algorithmic bias in automated transcription systems.[12]

Keywords: Speech-to-Text; Text Summarization; Smart Notetaking; Artificial Intelligence; Natural Language Processing; Deep Learning; Automatic Speech Recognition

How to Cite This Article

Pandey, S., Raj, P., Dutta, R., Mulla, E., & Medhe, V. (2026). Speech-to-text summarization system for smart note-taking. *International Journal of Advanced Scientific Research and Engineering Trends*, 10(1s), 234–245.

Introduction

The Historical Trajectory of Automatic Speech Recognition

The quest to develop computational systems that can learn and transcribe speech from human users has been ongoing for over 70 years, representing a dramatic paradigm shift from relatively primitive analog room sized machines to the highly optimized end-to-end deep neural networks running on resource constrained edge devices.[1] The origins of Automatic Speech Recognition (ASR) can be traced back to Homer Dudley of Bell Laboratories in the 1930s when he proposed a generalized framework for the analysis and synthesis of speech, paving the way for all acoustic processing technologies.[1] Nonetheless, the first significant milestone in the field was achieved in 1952 with Bell Laboratories researchers (notably H. K. Davis) developing the "Audrey" system, an acronym for Automatic Digit Recognizer, which was a specialized resonant acoustic machine that was able to recognize the utterances of all the electrifying countries in the world,1 calling out the digits zero to nine. After analysing the formants, or lowest resonant frequencies of the human vocal tract, the system achieved greater than 90% accuracy on the the original/primary speaker but suffered some degradation when tested on others and weighed 367kg in size![1]

Ten years later at the 1962 Seattle World's Fair, IBM unveiled a machine dubbed "Shoebox" that pushed the boundaries of the acoustic vocabulary to sixteen words of spoken English, including commands for "plus," "minus," and "total." Shoebox was then able to interface physically with a standard addition machine and perform the calculation of live speech-based equation input, truly the first voice-activated workflow.[16] Over the rest of the 1960s, international research institutions tore through the phonoma that make up our own heard language. Researchers in Japan created hardware that could distinguish one to six vowels and one hundred monosyllables of Japanese, while University College London made systems for distinguish four vowels and nine consonants from phonemes.[1] However, despite these advances in hardware, further loss was nigh impossible without sufficient mathematical model capabilities to tame the infinite train of spoken human language.[14]

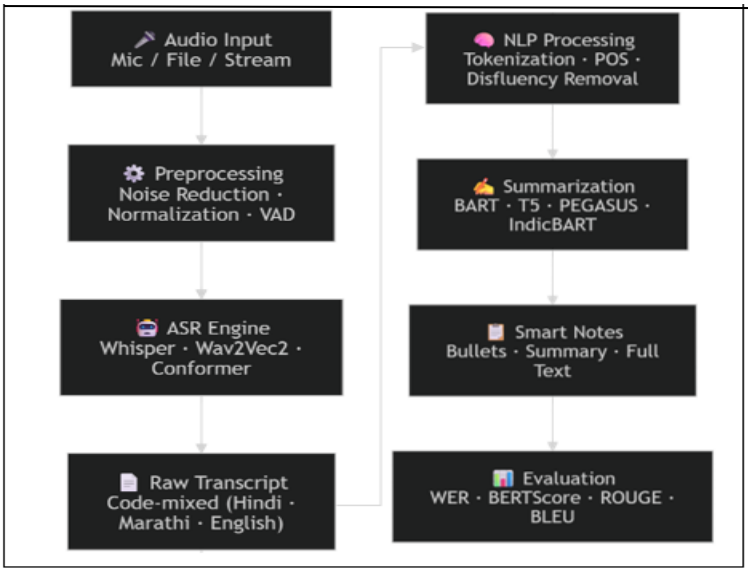

Fig. 1. *Speech-to_text Summarization Pipeline*

The mid and late 1970s was a time of rapid and generously funded progress, largely supported by the U. S. Department of Defence under the DARPA Speech Understanding Research (SUR) program, which ran from 1971 to 1976.[15] This virtually utopian goal was to develop a system that could understand and transcribe normal conversational speech, suitable for use in military and civilian fields. The most significant work of this period was the Carnegie Mellon 1976 system, "Harpy."[15] Harpy recognized 1,011 words (representing a child's working vocabulary at the age of three) and could recognize entire sentences by exploiting a beam search, in which each word was mapped onto a lexical graph, against a finite-state model of valid sentence structures.1 Around the same time, rule-based systems such as Hearsay[Moore et al. 1979] began to crack continuous speech, but continuing to rely on hand authored rules and comparatively inflexible templates.[17]

This led to an important paradigm shift in the 1980s from overly rigid template-matching techniques to flexible statistical models, primarily through the application of Hidden Markov Models (HMMs).[1] This represented a fundamental change in the way ASR systems worked by significantly increasing the efficacy of the computer to cater for the inherent variability in human speech. By applying intricate probability models, rather than rigid templates, HMMs could predict the most likely phoneme sequence from the observed acoustic signal. This advancement in statistical modelling, along with the advent of the n-gram language model, saw ASR vocabularies jump from a few hundred

words to tens of thousands and finally realized the commercialisation of ASR technology.[14] The first large vocabulary, continuous speech recognition system was launched by IBM in 1996, called MedSpeak, and was intended for large scale transcription of the more demanding lexicon of medical dictation.1 During this era, the HMM/ GMM system dominated, with the GMM decomposing the acoustic emission probabilities of the hidden state.[18]

The century changed from limited advances in the 20th century, where pace of improvements aligned with gains in computational power, enabling software companies to embed primitive, isolated speech recognition capabilities in commercial OS's.1 Then, in the late 2000s, the ascendancy of deep neural networks (DNNs) in the late 2000s revolutionized ASR, allowing researchers to better harness deep learning algorithms to accurately extract highly complex and non-linear acoustic features from data than traditional GMM systems were able to.[18] In the early days of hybrid DNN-HMMs, neural nets were used as a classifier to provide posterior estimates of state-tying hidden Markov models (HMM) which served as a temporal decoder, replacing the set of straightforward GMM emission probabilities.[18] As hardware and dataset sizes grew exponentially larger, so did the applicability of fully connected end-to-end neural architectures that completely dispensed with the HMM component of the solution. Currently, the contemporary ASR environment is largely governed by self-supervised Transformer variants and enormous encoder-decoder architectures pre-trained on gargantuan sums of untranscribed audio which can rival or surpass average human transcription performance in standard conditions.[1]

Evolution of Text Summarization Paradigms

Text summarization, strongly related to the latest breakthroughs in speech recognition technology, has evolved from a sparse statistical approach to a comprehensive understanding of the generative process. The main aim of text summarization is to generate a concise version of a source document while maintaining the most important information, accuracy, and meaning.[1] The first known formal publication in this area came in 1958 from Hans Peter Luhn, an IBM researcher who applied a statistical method to extract key sentences based on the frequency and placement of individual words.[2,3] Luhn's heuristic was based on the assumption that the main topics in a text would be the most frequently occurring words in the text.[1] This empirical method gave rise to the birth of computational linguistics and a series of rule-based feature engineering.[2] Edmundson extended Luhn's algorithm in 1969 by incorporating cue words ("in conclusion," "significant," etc.), sentence position algorithms (more weight for sentences at the top or bottom), and matching the title. These elements are then combined into a linearly weighted final score. This algorithm became the standard for extractive summarization for many years.[1] The extractive summarization automatically identifies, scores, selects, and then extracts sentences or phrases from a document and joins these together into a new document.[19] This method is computationally efficient and effective for information retention. However, there are two main drawbacks: poor readability and lack of natural flow. This results in a jumbled mess of mostly nouns as sentences are continually snipped out of a document.[1] The inability to express results clearly led to abstractive summarization. This method attempts to mimic human editing processes by understanding semantic meaning and producing new sentences or words to express these meanings in a way that is equivalent to the original document. This is a complex natural language generation problem that was beyond computational capabilities in the mid-20th century. In the 1970s and early 1980s, abstractive summarization used rule-based algorithms and knowledge bases for hierarchical propositional networks and rhetorical structure theory for document parsing and structuring.[20] In the case of RST, the texts were segmented with hierarchical relations between them labeled as either 'nucleus' or 'satellite.' The problem with these approaches was that they were computationally expensive and required extensive manual engineering, which was not feasible at the time in terms of the technology used.[1]

Deep learning methods were introduced in the early 2010s with the advent of the seq2seq model. Neural nets were able to "learn to map input word sequences to generated output word sequences."[2] Although these advances were significant in the field of abstractive summarization, recurrent nets still suffered with the hallucination of unfaithful, repetitive texts with the occurrence of long-range phenomena in the documents being summarized. It was not until the advent of the transformer and the LLM that abstractive summarization was able to reach the fluency required to create highly contextualized document synopses that are the core of the current LLM-based applications.[1]

Transformer Theory and Math

The transformer was first introduced in 2017 by Vaswani et al. It was the beginning of the end of the traditional sequential models in the fields of NLP and ASR.[1] The transformer was the first DL model that did not rely on the recurrent or convolutional neural nets that were used in the early days of deep learning. The traditional DL models used in the past were either recurrent or convolutional nets. Recurrent nets process the input in the sequence of the input tokens or frames, either in one or two directions.[21] The problem with the recurrent nets is the sequential constraint in processing the input data, which is the major bottleneck in the computation process.[21] The transformer overcomes the recurrent constraint by using the self-attention mechanism in processing the input data.[21] The self-attention mechanism is the core of the transformer architecture. It is the process of mapping the input sequence into the intermediate sequence using the three weight matrices Q, K, and V. Each feature is linearly transformed into these three intermediate spaces. The dot product of Q with the concatenation of K is taken over the input sequence. The dot product is then scaled by the square root of the parameter. The SoftMax is applied to the scaled dot product, and the resulting vector is the dot product of the SoftMax with V. The mathematical representation is as follows:

$$PE(pos, 2i+1) = cos(pos \ / \ 10000\hat{}(2i/d\_model))$$

Where dk is the dimension of the key vectors. This scaling factor is extremely important. Without it, the dot products could grow very large in magnitude, causing the SoftMax function to have very small gradients and making the model difficult to train.[23] To enable the model to jointly attend to information from different representation subspaces at different positions, the Multi-Head Attention architecture implements multiple attention functions running in parallel. This is done by linearly projecting the queries, keys and values h times with different, learned linear projections. The attention function is then applied in parallel to each of these projections, and the dv-dimensional output values are concatenated and projected again.[22] Mathematically this is given by;

$$MultiHead(Q,K,V) = Concat(head_1,\ldots,head_{\mathrm{h}})\, W^0$$

Where each,

$$head_{\mathrm{i}} = Attention(QW_{\mathrm{i}}^{Q}, KW_{\mathrm{i}}^{K}, VW_{\mathrm{i}}^{V})$$

And the Weight matrices WiQ, WiK, and WiV, are trained.[22] After the attention layer, information is fed into a position-wise FFN to normalize the encoding. This takes the form of two linear transforms with a ReLU between them:[22]

Since the Transformer does not operate sequentially, it totally depends on explicit positional encodings to be added to the input embeddings to give it some information about the relative position of tokens in the sentence.1 The following section describes how the positional information is inserted using sinusoidal functions of different frequencies. For position and dimension, the positional encoding is defined as:

This particular geometric sequence of wavelengths, 2π 1000.2π, that the model can attend to over at each layer, is convenient for the model to attend to by relative position as it can learn trivially how to weight it accordingly. Since the derivative of sin(x) is cos(x) and vice versa, this design further ensures that gradient flows are extremely stable over this large range of positions and dimensions and is consequently essential for training very deep neural networks.[24] By choosing the denominator as $10000^{2i/\mathrm{dmodel}}$, the rate of change of the sinusoid is moderated.[24] The nature of this attention technique is highly GPU-friendly and this allows the architecture to be trained efficiently on a large amount of data on current GPU hardware and reduces the amount of time taken to train the model while allowing the size of the model to scale to billions of parameters leading to the successful implementation of many state-of-the-art large language models and acoustic transcribers.[21]

Advanced Acoustic Modelling: Whisper, Wav2Vec 2.0, and Conformer

Creating an effective Speech-To-Text summarization system relies on a strong acoustic model that can handle various accents, background noise, and fast speech.[1] Traditional hybrid systems trained separate models for acoustic signals, pronunciation, and language. Modern end-to-end architectures combine these steps, turning spectrogram inputs directly into text tokens. Leading frameworks in this area come from OpenAI, Meta, and Google: Whisper, Wav2Vec 2.0, and Conformer.[1]

OpenAI's Whisper represents a major shift by using extensive supervised training. It's an encoder-decoder Transformer trained on 680,000 hours of multilingual, multitask audio data collected online.[1] Unlike many top models that rely on self-supervised pretraining due to limited labeled data, Whisper pushes supervised learning to an extreme. It processes audio in 30-second segments converted into log-mel spectrograms. Earlier models used 80 mel frequency bins, but the large-v3 version uses 128 for finer frequency detail.[25] The spectrogram is passed through a "conv stem" of two convolution layers with GELU activations to shorten the sequence and match the Transformer's hidden state.[25] Positional encoding is added before feeding into deep self-attention and MLP layers in the encoder.

The decoder generates text autoregressively. Whisper's key innovation is a multitask training setup: special tokens direct it to handle tasks like language ID, timestamps, transcription, and translation. This single model can perform many functions, showing strong zero-shot ability. It often makes about 50% fewer errors than specialized systems, especially with technical jargon, strong accents, and noisy backgrounds.[42]

Wav2Vec 2.0 takes a different approach, relying on self-supervised learning to build powerful speech features from raw, unlabeled audio. Its architecture has a multi-layer convolutional encoder that extracts continuous latent speech representations from normalized audio waveforms.[26] Applying Transformers to raw audio is challenging since the data is continuous. Wav2Vec 2.0 addresses this by using a quantization module that converts these latent features into a limited set of discrete targets.[1] During pretraining, it masks parts of the latent sequence by randomly selecting start points with a certain probability (e.g., 6.5%) and masking a fixed length (e.g., 10 units).

Successive time steps from each index.[44] The contrastive loss uses the InfoNCE loss and trains the Transformer context network to select the appropriate quantised representation for time-step t, which is masked, out of a set of K distractors sampled from other hidden time steps within the same utterance.[26] This loss can be formulated as:

$$L_{\mathrm{m}} = -log\ (\ exp(sim(c_{\mathrm{m}}, q_{\mathrm{i}})/k)\ /\ \Sigma\ exp(sim(c_{\mathrm{m}}, q_{\mathrm{j}})/k)\ )$$

Where sim is the cosine similarity and is the temperature in the contrastive loss.[26] To allow for complete backpropagation when selecting discrete codebook entries, Wav2Vec 2.0 optimizes the Gumbel-Softmax trick. To encourage the model to select one cell of the codebook in the most greedy way possible, the Gumbel-Softmax temperature T is rapidly annealed during training from 2.0 down to a minimum of 0.5 for the BASE architecture, and 0.1 for the LARGE architecture which steadily reduces randomness and sharpens the codebook selection as training proceeds.[26] This careful process allows Wav2Vec 2.0 to learn extremely strong acoustic representations, surpassing most supervised speech models when fine-tuned on extremely scarce labeled data, making it a very attractive model to explore for low-resource linguistic tasks.[1]

Table 1. *Comparison of Speech Recognition Model Architectures*

Model Architecture	Training Paradigm	Core Mathematical Mechanism	Key Acoustic Strengths & Innovations
Whisper (OpenAI)	Fully supervised	Encoder–decoder transformer with log-Mel convolutional stem	Unmatched robustness to noise, multilingual translation via special tokens, strong zero-shot capability
Wav2Vec 2.0 (Meta)	Self-supervised	CNN feature encoder + Gumbel-Softmax quantization with contrastive learning	High transcription accuracy with minimal labeled data using InfoNCE masking loss
Conformer (Google)	Supervised / hybrid	Convolution-augmented transformer with Macaron-style feed-forward blocks	Captures local acoustic patterns and global context efficiently

Conformer: The Conformer (Convolution-Augmented Transformer) aims to merge concepts: mathematically fusing the extant, huge-Transformer's global context modeling ability with the localized, fine-grained feature extraction powers of CNNs.1 Traditional global self-attention often neglect or over-smooth fragile phonetic transitions in fast conversational speech.1 The insertion of depthwise convolution modules into the model architecture allows capturing important local acoustic patterns.5

The architecture is defined by the unique "Macaron-net" sandwich structure. Unlike the single feed-forward network following the attention mechanism in a standard Conformer Block, the Conformer block use two half-step feed-forward layers, one right prior to the multi-head self-attention and one right after the convolution module 5 xi is fed into the the two half-steps feed-forward layers as follows:[2,3]

$$x_i^n = x_i' + Conv(x_i')$$

$$x_i' = x_\mathrm{i} + MHSA(x_\mathrm{i})$$

$$y_\mathrm{i} = LayerNorm(x_i^n + 1/2\, FFN(x_i^n))$$

Each feed-forward module utilizes an expansion factor of 4, Swish activation functions, and pre-norm residual units.[5] Extensive ablation studies have repeatedly demonstrated that this Macaron-style sandwiching provides a highly significant improvement over architectures utilizing a single feed-forward module, particularly in challenging ASR niches such as accurately recognizing the erratic pitch and tone of children's speech.[5]

Abstractive Summarization Engines: T5, BART, and PEGASUS

Following the precise transcription of speech to text, the crucial secondary objective of the application is the automated generation of a concise, highly coherent summary. Transformer-based models have thoroughly revolutionized this specific task, with BART, T5, and PEGASUS emerging as the premier, highly competitive architectures, each utilizing distinct pre-training paradigms.[1]

BART (Bidirectional and Auto-Regressive Transformer): BART utilizes a remarkably standard sequence-to-sequence Transformer architecture, conceptually and mathematically generalizing both BERT (due to its bidirectional, context-aware encoder) and GPT (with its left-to-right, autoregressive decoder).[1] BART features approximately 10% more parameters than a comparatively sized BERT model, initializes its parameters from a normal distribution N(0, 0.02), and utilizes the GELU activation function.[45]

Crucially, it omits the additional feed-forward network typically found at the top of BERT before word prediction.[45]

Table 2. *Comparison of Text Summarization Model Architectures*

Summarization Engine	Core Architectural Premise	Pre-Training Objective / Noising Function	Primary Use Case Strengths
BART (Meta)	Generalizes BERT encoder and GPT decoder	Denoising autoencoder (infilling, deletion, permutation)	Highly effective for conversational dialogue, tone preservation, and dynamic response

			generation
T5 / mT5 (Google)	Unified text-to-text framework	Span corruption (predicting masked sentinel tokens)	Highly versatile across NLP tasks; mT5 supports 100+ languages with large vocabulary coverage
PEGASUS (Google)	Dedicated summarization architecture	Gap sentence generation (GSG) using ROUGE-based sentence selection	Excellent at extracting key factual content from long, formal documents

BART is pre-trained exclusively as a denoising autoencoder. The intensive training process involves intentionally and severely corrupting the input text and subsequently training the neural model to flawlessly reconstruct the original, uncorrupted document.[46] This specific mathematical objective forces the model to deeply understand complex syntax. BART utilizes five distinct arbitrary noising functions:

- Token Masking: Random tokens are replaced with `` elements.
- Token Deletion: Random tokens are deleted entirely, forcing the model to deduce which positions are missing inputs.
- Text Infilling: Entire spans of text of varying lengths are replaced with a single mask token, teaching the model to predict how many tokens are missing.
- Sentence Permutation: Documents are divided into sentences based on full stops and shuffled randomly.
- Document Rotation: The document is rotated to begin at a random token, training the model to identify true document boundaries.[45] Empirical evaluations demonstrate that BART consistently achieves state-of-the-art results on dynamic conversational response generation, exhibiting exceptionally high precision in transferring factual information without hallucination.[48]

T5 (Text-to-Text Transfer Transformer): T5 takes a highly generalized approach, treating absolutely every Natural Language Processing task—whether language translation, sentiment classification, or complex abstractive summarization—strictly as a unified "text-to-text" problem.[1] It requires the raw input to be explicitly formatted with a specific, rigid task prefix (e.g., appending "summarize: " before the document).[47] T5 is pre-trained using a highly effective span-corruption objective, where consecutive spans of text are replaced with a single, unique sentinel token, and the model must accurately predict the missing spans.[1]

While the standard T5 is incredibly powerful for English, adapting it to multilingual environments necessitates the mT5 variant. To effectively process over 100 languages, mT5 dramatically expands its SentencePiece vocabulary size to approximately 250,000 subwords, ensuring high coverage across disparate orthographies.[48] However, this bloated vocabulary exponentially increases the model's parameter count. The embedding matrix alone causes the mT5-Base model to swell to 580 million parameters.[48] While mT5 is highly flexible and records exceptional ROUGE metrics across diverse datasets, its massive size severely hampers its deployment on constrained computational infrastructure, and in highly casual conversational contexts, it occasionally hallucinates informal phrases compared to BART's tighter factual adherence.[48]

PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization): Unlike highly generalized models, PEGASUS was designed, architected, and pre-trained explicitly for the solitary downstream task of abstractive text summarization.[1] Its novel, highly specialized pre-training objective, known as Gap Sentence Generation (GSG), involves mathematically masking out entire, highly important sentences from an input document and forcefully training the decoder to autoregressively reconstruct those exact gap sentences entirely from the remaining context.[58] The underlying hypothesis is that if a model can accurately predict the most critical sentences missing from a document, it has effectively learned the core human mechanics of summarization.[58]

The selection of these gap sentences is a meticulously calculated process. Rather than masking sentences randomly, PEGASUS evaluates sentence saliency by computing the ROUGE1-F1 score between each individual sentence and the remainder of the document.[6] The sentences achieving the highest "principal" scores are selectively masked.[49] This tight alignment of the pre-training mechanism with the downstream summarization task significantly enhances its performance during fine-tuning. PEGASUS excels at rapidly generating succinct and highly insightful summaries for lengthy scientific publications or structured news articles, routinely achieving human-level quality with as few as 1,000 fine-tuning examples.[48] However, it occasionally struggles with the non-linear narrative structures of messy, multi-speaker conversational datasets.[1]

Linguistic Challenges, Code-Mixing, and Multilingual Processing

While large models perform well with high-resource languages like English, applying them to linguistically complex regions such as India brings greater computational, phonological, and morphological challenges. India is very diverse linguistically, with 22 scheduled languages across distinct language families. The major ones are Indo-Aryan languages like Hindi, Marathi, and Bengali, and Dravidian languages such as Tamil and Telugu.[1]

A key issue in Indian language ASR and NLP is data scarcity. Most regional languages lack large, high-quality parallel corpora, curated text collections, or thoroughly transcribed speech datasets needed for data-intensive deep learning. Moreover, Indian languages often have complex morphology and syntax.[1] For example, Hindi and Marathi show rich inflection and agglutination, where a single root can generate many forms based on gender, number, tense, and case. This complexity greatly increases vocabulary size during language modeling, demanding much larger training data. Phonologically, Hindi features diverse accents across 18 states and 85 districts, causing computational difficulties as similar phonetic patterns appear with varying speech rates.[27]

A prominent sociolinguistic trait is code-switching, widespread across India. In casual conversation, formal settings, or lectures, English words are often mixed into bilingual sentences, creating hybrids like Hinglish.[1] This rapid language swapping disrupts the acoustic and syntactic assumptions of monolingual models, leading to heavy cross-lingual interference.[2] ASR systems trained only on Hindi fail to transcribe such mixed speech accurately, resulting in high error rates.[1]

To measure code-mixing complexity, computational linguists apply metrics like the Code-Mixing Index (CMI). This index quantifies the proportion of words not in the primary (matrix) language. For an utterance with n total tokens and u language-independent tokens (such as named entities, punctuation, numbers), CMI is calculated accordingly:

$$CMI = 100 \times [1 - max(w_i) / (n - u)]$$

Where max(wi) is the number of token in the most dominant language.[28] A CMI of zero signifies strict monolingualism while a large CMI value would suggest heavy code-mixing.[28] Formulations of the CMI take into consideration the number of instances of switches within the sequence i.e. switch points, it also takes into account the syntactic movement within the discourse.[50] The M-index, based on the Gini coefficient, is a score used to measure the inequality of the distribution of language tags in a corpus ensuring that the training data had sufficient language balancing to ensure that the neural network would not catastrophicically forget the low resource language while learning to process code-switched speech while being trained on a primarily high resource language.[28]

The AI4Bharat Ecosystem and Dataset Engineering

In order to systematically alleviate the extreme computational problems posed by data scarcity and immense linguistic diversity, large research efforts including AI4Bharat painstakingly curated very fine-grained datasets and foundation models specific to Indian languages.[1]

In the acoustic and transcription domain, availability of good training data is crucial. The GramVaani ASR corpus is an important milestone in this regard, as a forerunner test bed for Hindi ASR in the wild. It has over 1000 hours of open domain spontaneous phone calls without annotation and 100 hours of human transcription from 132 speakers distributed over 82 districts.[27] Such a data set is designed to include every conceivable chaotic regional accent variations, crawling levels of background noise and in variety of natural code switching from rural India and thus be an immensely difficult, real-world data set which will unmask the brittle nature of sterile, studio-trained acoustic models.[29]

For the extremely challenging tasks of text generation and abstractive summarization, the AI4Bharat team effectively developed IndicBART, a cutting-edge multilingual seq2seq model that was heavily pre-trained on 11 different Indic languages together with English.1 The IndicBART model openly fills the yawning architectural gaps inherent to the “Massively Multilingual Massive” models such as mT5. Due to the sheer fact that models such as mT5 must try to cover more than 100 languages with a single model, they carry bloated, extremely inefficient vocabularies that are unable to accurately encode the microscopic morphology structures of specific low-resource Indian language families.1

Lingering inefficiencies are solved through a handsome architectural shortcut: orthographic script unification. Since most Indian languages are written in Abugida scripts echoing from ancient Brahmi, their logical and phonetic structures seem to converge on an extraordinarily high level1. Before passing to the model, the IndicNLP package first normalizes all texts to standardize use of Zero Width Joiners (ZWJ) and Nuktas (dependent vowel signs of all sorts),[30] then translates all Indic languages into the same unified scriptDevanagari8. This brilliant unification allows IndicBART to have a tiny requisite vocabulary size here 244 million parameters8 while still capturing the full knowledge required to state-of-the-art performance on Indian language summarization with 452 million sentences of training data.[8]

For these models to be evaluated in conversational settings, it is essential that domain-specific data be used. Summarization research has historically used mainly comparatively high-level, single-speaker texts, such as news articles (e.g., CNN/DailyMail).[31] When these models are tested on more haphazard, multi-party conversations, they notoriously hallucinate factual inaccuracies. Expanding the training and testing data to include conversational data is therefore crucial, which is where specialized benchmark corpora such as SAMSum are mandatory. SAMSum is a large (16,000+ speaker-like messenger chat conversation corpora produced entirely through professional linguists7), high-fidelity, colloquial chat Corpus, spontaneously including slang, code-switches, typos, and non-sequential stories,[34] with corresponding human-written, high-quality third-person abstractive summaries.[32] Using this sort of corpora to develop text-to-text summarization applications is therefore essential for being reliably scaled for multi-polar conversation.[33]

Evaluation Metrics: Quantifying ASR and Summarization Efficacy

A proper validation of the actual efficacy and Accuracy of the integrated Speech-to-Text application would involve rigorous, mathematical validation using super specific, domain-standard metrics that suitably compare and illustrate that human judgment1.

ASR Evaluation (wer and cer): The de-facto metric for measuring accuracy of Automatic Speech Recognition is the Word Error Rate (or WER). WER is a strict comparison of the automatic recognition output versus the human transcribed ground-truth reference document, calculated using the Levenshtein distance algorithm.[2] The below is how WER is computed from the recognition output:

$$WER = (S + D + I) / N$$

Where S is the number of absolute substitutions (words that are mis-recognized and substituted in the model output), D is the number of deletions (words that are entirely missed in the output transcript), I is the number of insertions (words that are hallucinatorily added by the model output) and N is the number of absolute words in the reference text.[10] WER may be scored as 0.0, which would identify a completely perfect transcription. Due to the inclusion of irrelevant insertions in the numerator, WER is also capable of scores greater than (1.0)100, identifying a perfect score with a useless system that hallucinates extensively in highly noisy environments.[1]

In such cases, WER now becomes more limited and, if the ASR is now set to run across multilingual sets with a highly agglutinative language with no lexical pauses in the set of words in a sentence (like Hindi or Maratha),[1] it is now that WER can fail in many different ways.[34] A single highly agglutinative string can be killed off by just one wrong character much down the entire word which can generate a whole zeros value. Thus the total C is unlikely to justify the true acoustic knowledge of the model.[10] This is why the use of Charater Error Rate (CER) is so often used in conjunction to measure the performance as it is used on the same principles of Levenshtein distance as WER though measures of differences on a purely alphabetic representation, and hence has a much finer grained performance valuation incorporating spellings, dialect differences, jargonisms and so on.[1]

Summarization Evaluation (ROUGE and BERTScore): the most established, historically standard measure of rigorously evaluating text summarization is the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric. ROUGE measures overall quality of a model-generated summary via reductive information retrieval by directly counting the exact number of shared n-gram overlaps with human referenced summaries.[1] ROUGE-N measures only n-gram overlap explicitly, where ROUGE-L measures the overall sentence structure via a Longest Common Subsequence (LCS) algorithm.[1]

But ROUGE suffers from astounding, built-in algorithmic disadvantages it‘s a fundamentally shallow lexical metric that measures only exact n-gram overlaps.1 Worst, it has built-in “semantic blindness” and simply isn‘t equipped to acknowledge true synonyms, let alone complex contextual paraphrasing.[36] A theoretically excellent abstractive system might produce an apparently perfect summary in an entirely different vocabulary one that surpasses the reference, and get a disastrous low score.[36]

To combat this embarrassing ground-truth error, more sophisticated neural evaluation measures such as BERTScore are often weighed more heavily by the research community.[1] BERTScore measures the exact cosine similarity of deep contextual embeddings taken from a pre-trained language model between the machine generated text and the reference text.[11] More specifically, the algorithm greedily matches each token of the reference sentence to the token in the generated sentence which results in maximum cosine similarity scores. These maximum scores are then summed and divided by sequence length in order to obtain a Recall, Precision, and F1 score.[11] For better accuracy, BERTScore optionally weights the individual word similarity scores by Inverse Document Frequency scores, such that truly rare, highly semantic words are weighted greater than common, stop words.[37] Since BERTScore intrinsically relies on these dense vector representations in lieu of surface strings of tokens, it effectively and consistently measures the semantic similarities and differences of strings, opening the floodgates for paraphrasing reward even with zero lexical overlap.[35]

Educational Impact, Cognitive Load, and User Experience Design

The adoption of pervasive automatic lecture summarization and meeting transcription services presents momentously important implications for the contemporary learning pedagogy, enterprise productivity scaling, and lifelong learning processes.1 In intense educational settings, pervasive speech2to3text summarization directly counteracts the crippling detriments of passive learning and permanently alleviates the overwhelming cognitive burden that is regularly encountered by students diligently participating in sophisticated, real-time audio lectures.[12]

Cognitive load theory rationalizes human working memory as having a endogenously restricted load limit for multiple types of concurrent processing,[13] referring to the overspill that indiscriminately occurs when an activity continues to demand cognitive resources beyond the absolute ceiling.[13] In reality, mentally following a dense technical lecture, unraveling orbits of intricate mechanisms, and eagerly compiling and transcribing personal longhand notes almost always push the total load imposed by faculty tasks above this permissible ceiling thus penalizing comprehension.[13] Referring to even the strongest form of emotional salience, peer-reviewed academic literature definitively supports that in-stream automatically generated, high fidelity briefs appearing alongside lecture video enhances productivity while

studying.[39] Freesing the extraneous load of transcripting allows students' limited load capacity to be dedicated to germane cognitive load the associations, the virtual circuits and the profound consequences not racing to record raw facts.[13]

Students exposed to this modality and AI-powered feedback routinely outperform other students watching bare video[39] Electronic summarization is inherently a super-effective pedagogical technology, allowing students to do rapid active review. AI generated summaries, emphasizing high-value fundamental concepts, wave away off-topic distractions. The student leverages the summaries as compact reference points, while inputting questions and note-ins to extend the focus and articulate a continuous examination loop.[38]

Because of how quickly omnipresent AI transcription tools are being developed and adopted, however, extremely rigorous UX design protocols are needed digital interfaces need to be made to fit into traditional professions' workflows in undetectably smooth yet still tightly visual ways so that users can verify the accuracy of information instantaneously.[1] The "human-in-the-loop" prediction paradigm is thus still crucial for the immediate and anticipatable prevention of hazardous model hallucinations and for maintaining stringent factual standards in the perilous professional, legal, and clinical Information[7] spaces where algorithmic errors may directly manifest as templatized permanent electronic records.[13]

Ethical Considerations and Privacy by Design

Finally, the widespread, often invisible adoption of continuous recording applications introduces exceptionally severe ethical, regulatory, and privacy considerations that must dictate system architecture.1 Captured audio data processed by third-party transcription tools contains vastly more than just the literal transcribed spoken words; human speech inherently carries highly sensitive "voice-inferred information".[40] This encompasses deeply non-obvious biometric identifiers, emotional states, physical health markers, socioeconomic demographics, and unique vocal tract resonances that can easily facilitate unauthorized re-identification.13

The increasingly standard corporate practice of deploying AI note-takers in virtual meetings frequently occurs without the explicit, universally informed consent of all active participants, severely risking unauthorized corporate surveillance and devastating identity theft through potential data breaches.1 In many software ecosystems, a single meeting host can delegate an AI agent to record and transcribe a session without any affirmative action from other participants.[41] Under strict regulatory frameworks such as the European General Data Protection Regulation (GDPR), consent must be informed, specific, and freely given; delegating consent to a single participant fundamentally violates these tenets and exposes deploying organizations to massive legal liability.[41] Furthermore, the persistent storage of identifiable speech data on third-party cloud servers outside the direct control of the user raises severe questions regarding data minimization, cross-border transfers, and algorithmic monetization.13

Consequently, software developers engineering Speech-to-Text Summarization applications must rigorously implement strict privacy-by-design architectures. This necessitates the creation of crystal-clear consent notice mechanisms, the execution of robust end-to-end data encryption protocols, and the enforcement of fully transparent data retention policies to thoroughly protect the highly sensitive, biometric nature of processed human speech from malicious extraction.[1]

System Design and Computational Details

Project Overview & Aim

The primary objective of the architecture is to engineer a highly accurate, end-to-end applied deep neural network pipeline that combines Automatic Speech Recognition (ASR) and Abstractive Text Summarization. The distinctive technical footprint of this system lies in its robust capability to handle structurally chaotic conversational contexts, aggressive code-mixing (such as continuous English-Hindi insertions), and morphologically complex, low-resource Indian languages (Hindi and Marathi).

Data Ingestion & Preprocessing Pipeline

The system relies on a rigorous data pipeline formulated across three main processing environments:
ASR Data Preprocessing (LibriSpeech & IndicSpeech):
Data Aggregation: Handled via dedicated audio pipelines that process both English and Hindi audio sets. Audio is resampled and localized systematically into sorted directories (english_audio, hindi_audio).
Text Normalization: Utilizes regex rules for strict punctuation/special character removal and unicodedata libraries to standardize deep acoustic anomalies and varied character encoding across Indic scripts.
English Conversational Summarization (SAMSum Corpus):
Data Anatomy: ~14,000 multi-speaker dialogue-summary pairs structured closely akin to chat messenger platforms. Average dialogue length sits at roughly 90 words with 15-word summaries.
Outlier Removal: The pipeline forcefully enforces a max threshold limiter (512 token cutoff), filtering extremely long anomalies to prevent computational bottlenecks during Transformer encoding and to guarantee a balanced input layer.

Hindi Text Summarization (IndicCorp/Hindi News):
Data Anatomy: A massive corpus of ~260,000 Hindi article-headline pairs (average 362-word articles to 11-word headlines).
Noise Reduction: Specifically strips empty and highly trivial content sets (<30 words), pushing only highly contextual arrays forward to prevent severe model hallucination on sparse inputs.

Subsystem A: Acoustic Modeling & ASR Engine

The application transitions raw, continuous, unbroken analog audio into discrete, structured textual data. The architectural selection for this layer involves deep evaluation of three distinct state-of-the-art frameworks to overcome acoustic constraints:

OpenAI's Whisper (Supervised Paradigm): Selected for its unmatched zero-shot generalization capabilities. It maps log-mel spectrograms (128 frequency bins in large-v3) compressed through a specialized 1D convolutional stem. It is deeply effective in tackling abrupt Hindi-English code-switching and noisy dialectal variance.

Wav2Vec 2.0 (Self-Supervised Quantization): Designed to overcome the core problem of Indian language data scarcity. Rather than relying entirely on supervised labels, it utilizes a multi-layer CNN feature encoder combined with a Gumbel-Softmax trick to quantize continuous audio representation.

Conformer (Macaron-net Architecture): Utilizes dual half-step "Macaron" structures sandwiching depthwise convolution modules natively inside a Transformer block. Strongly prioritized for capturing fine phonetic transitions often lost in global self-attention during rapid conversational speech.

Subsystem B: Abstractive Text Summarization Engine

Upon generating the textual transcript, the system transitions into contextual Natural Language Generation (NLG) aiming at semantic comprehension over rigid rule-based text extraction.

BART & T5 Arrays (Conversational Processing):
BART is geared as a Denoising Autoencoder using token masking, deletion, and rotation, heavily advantageous for accurately re-aligning fragmented conversational structures (like SAMSum scripts).
T5 (mT5 Variation) is utilized for unified text-to-text language crossing but bears a heavy parameter load computationally due to its 250,000+ token vocabulary limit.
PEGASUS: Engineered primarily for formal, long-document summarization leveraging its Gap Sentence Generation (GSG) mathematics, utilizing principal ROUGE overlap pre-training techniques.
Highly Specialized IndicBART (AI4Bharat Ecosystem): The most critical piece for the Hindi/Marathi processing layer. Recognizing the computational bloat in mT5, IndicBART deploys Devanagari orthographic script unification—converting and normalizing mathematically shared linguistic traits of Indo-Aryan scripts. This structurally shrinks the model size to an efficient 244-Million parameters whilst producing state-of-the-art inference results inside Indian language ecosystems.

System Evaluation & Performance Metrics

To bypass the limitations of superficial performance evaluations, the system leverages a dual-layer metric testing environment:

ASR Analytics (WER & CER):
Word Error Rate (WER): Standard Levenshtein calculation measuring swaps, insertions, and hallucinations.
Character Error Rate (CER): Heavily preferred in this system's context. Since Hindi/Marathi are agglutinative languages lacking pure word boundaries, CER avoids throwing false negatives out on minor sub-word inflection errors.
Summarization Analytics (ROUGE & BERTScore):
ROUGE: A baseline statistical heuristic check for n-gram extraction overlap.
BERTScore: Calculates the dense spatial cosine similarity of contextual embeddings. This validates the system's ability to abstract, paraphrase logically, and retain high factual integrity across generated responses, totally eliminating semantic blindness found in rigid lexical checkers.

Non-Functional Requirements & Socio-Technical Implementations

UX & Educational Cognitive Load: Built explicitly to transfer processing tasks, minimizing split-attention constraints by preventing users from having to rapidly capture complex spoken dialogue, thus increasing germane learning capacities significantly.

Privacy-By-Design Protocol: Addressing the inherent dangers of retaining voice-inferred biometric signatures via constant third-party conversational transcription. The design actively governs user consent tracking frameworks, strictly minimizing third-party retention, and securing conversational payloads with transparent localized handling limits to mitigate surveillance liabilities.

Conclusion

The confluence of Automatic Speech Recognition and abstractive text summarization is a winning stride for the modern era of information consumption as it democratizes the transition from unwieldy oral minutiae to granular, prescriptive text in a fully automated manner. This entire paper has highlighted the fact that while self-supervised giant models like Whisper epitomize the state-of-the-art in a resource-rich setting, extremely specialized self-supervised modeling infrastructure like Wav2Vec 2.0 and IndicBART are equally essential to unravel the phonetic and morph-memic riddles posed by the low-resource Indian tongues Hindi and Marathi. The careful choice of leveraging Devanagari orthographic unification, sophisticated Gumbel-Softmax vector quantization, and embeddable edge Conformer blocks systematically mitigate the historical limitations of data scarceness and rampant codeswitching. The robust solutions achieved for a wide array of hands-on tasks are also reflected in the fact that accurate system evaluation must rely on a transition from static lexical methods like ROUGE to deeply semantic metrics like BERTScore to capture conversational context. As these systems quickly become an integral part of Education methodologies to reduce working memory overheads and bring about automation to Content Management markets, it will be vital to adhere to stringent data privacy frameworks and coupled informed consent-driven end-user design guidelines. Finally, the marriage of accurate acoustic models with state-of-the-art NLP generation will change the very fabric of knowledge sharing across language barriers.

Acknowledgement

The successful culmination of this final year project, titled "Speech to Text Summarisation App," would not have been possible without the invaluable guidance, academic support, and resources provided by the Department of Computer Engineering. Sincere and profound gratitude is extended to the esteemed Project Guide, Prof. Mrs Vidya Medhe, whose technical insights and continuous encouragement shaped the theoretical and practical dimensions of this work. Deep appreciation is also expressed to the Head of the Computer Engineering Department, Prof. Mr Sandip Vanjale, for fostering a rigorous academic environment conducive to advanced computational research. Furthermore, profound thanks are offered to the Principal of Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, Dr. Rajesh Prasad, for providing the institutional infrastructure and visionary leadership that facilitated this endeavor. The collective support, constructive feedback, and expertise of all associated teaching and non-teaching supporting faculties have been immensely instrumental in realizing this academic pursuit.

References

1. Baevski, A., et al. “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations.” *NeurIPS*, 2020.
2. Radford, A., et al. “Robust Speech Recognition via Large-Scale Weak Supervision.” *OpenAI*, 2022.
3. Raffel, C., et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5).” *JMLR*, 2020.
4. Zhang, J., et al. “PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization.” *NeurIPS*, 2020.
5. Wolf, T., et al. “Hugging Face Transformers Library.”
6. Lhoest, Q., et al. “Hugging Face Datasets Library.”
7. Kaggle. “Public Machine Learning Datasets.”
8. Panayotov, V., et al. “LibriSpeech: An ASR Corpus Based on Public Domain Audio Books.” *IEEE*, 2015.
9. GramVaani. “Hindi Speech Corpora Dataset.”
10. Bird, S., Klein, E., and Loper, E. “Natural Language Processing with Python.” O’Reilly Media.
11. Vaswani, A., et al. “Attention Is All You Need.” *NeurIPS*, 2017.
12. Brown, T., et al. “Language Models are Few-Shot Learners (GPT-3).” *NeurIPS*, 2020.
13. Mozilla Foundation. “Common Voice Dataset.”
14. Python Software Foundation. “Python Documentation.”
15. Multilingual ASR. “ResearchGate Publication.”
16. Baevski, A., et al. “wav2vec 2.0 Overview.” Web article.
17. OpenAI. “Introducing Whisper.”
18. PEGASUS. “Medium Article.”
19. IndicBART. “Hugging Face Model Repository.”
20. CER Metric. “Galileo AI Documentation.”
21. UCSB. “History of Automatic Speech Recognition.”
22. US Legal. “ASR Historical Development.”
23. Wikipedia. “Speech Recognition.”
24. ClearlyIP. “Voice Recognition History.”
25. Wikipedia. “Automatic Summarization.”
26. arXiv. “Abstractive Summarization Research Papers.”

27. ResearchGate. “Comparison of BART, T5, and PEGASUS Models.”
28. mT5. “Medium Article.”
29. PEGASUS. “Hugging Face Repository.”
30. IEEE. “Code-Switching in Hindi-Marathi Speech.”
31. ISCA. “Whisper for Hindi Code-Mixed Speech.”
32. MUCS 2021. “ResearchGate Publication.”
33. Skit Tech. “Code Mixing Metrics.”
34. GramVaani. “Speech Dataset.”
35. OpenSLR. “1111 Hours Hindi ASR Dataset.”
36. AI4Bharat. “IndicBART Model.”
37. Indic NLP Library. “Open Source Toolkit.”
38. GeeksforGeeks. “Indic NLP Overview.”
39. Hugging Face. “SAMSum Dataset.”
40. Zhang, T., et al. “BERTScore: Evaluating Text Generation.” *arXiv*, 2019.
41. Emergent Mind. “BERTScore Overview.”
42. arXiv. “Code-Switching ASR Research.”
43. Hugging Face. “Whisper Large-v3 Model.”
44. arXiv. “Survey on Abstractive Summarization.”
45. Bambal, M. “Understanding Whisper's Encoder–Decoder Transformer Architecture.” Medium.
46. “A Comparative Analysis Between Conformer-Transducer, Whisper, and wav2vec2 for Child Speech Recognition.”
47. Rastogi, R. “Papers Explained: wav2vec 2.0.” Medium.
48. “Wav2Vec2: A Self-Supervised Learning Technique for Speech Representations.”
49. “Brief Review: Conformer – Convolution-Augmented Transformer for Speech Recognition.”
50. anwarvic. “mT5: Multilingual T5.