AF-HPSum: Agentic Framework for Multi-Strategy Summarization of Hindi Podcasts

Anonymous ACL submission

Abstract

Podcasts are lengthy audio conversations which require a significantly large context input for accurate summarization via computational models. Despite recent advancements in Large Language Models (LLMs), it is challenging to summarize a transcribed podcast conversation using LLMs, due to input context length, long-range dependencies, noisy data and attention mismatch. In this paper, we propose an agentic framework for LLM-based summarization of Hindi Podcasts (**AF-HPSum**) which leverages multiple strategies, including a rule-based deletion strategy for compressive summarization. Using multiple LLMs, both open-weighted and closed-source, we evaluate the performance of our framework and observe that an iterative strategy helps preserve long-range dependencies and produce relevant summaries. We also conducted a preliminary human evaluation, which elicits model selection and helps build a comprehensive pipeline for podcast summarization. Through parameter-efficient training of open-weighted models and our iterative approach, we achieved a significant performance improvement over closed-weight and larger models by a significant margin. We will release our framework codebase, prompts, data and output with this paper here.

1 Introduction

Automatic summarization as a field has made rapid progress in recent times since the Transformer architecture (Vaswani et al., 2017) and Large Language Models (Raffel et al., 2019) have been developed. Since then, numerous models for general tasks as well as some finetuned for summarization have been developed. While a lot of work has already been done to enhance the capabilities of these models in the English language, Indian languages like Hindi, Marathi, Bengali and various others are lagging behind in terms of understanding of the languages themselves. In this paper, we will specifically discuss approaches to summarize text in Hindi language, taking Podcast data as an example.

Podcasts (Karbalaee, 2023) are a form of digital audio entertainment that cover a variety of topics like stories, debates, interviews, narratives covering various genres, etc. The number of such podcasts online is huge, with new ones arriving daily in the dozens. Podcast descriptors or summaries are a way of getting a brief idea of the content of a podcast and to draw viewership. Such podcast descriptors are usually written by the authors themselves or by the hosting platforms and are quite unreliable, they may sometimes exaggerate or are completely unrelated to the main topic of the podcast. Through our research we will show that such summaries are less liked by readers compared to LLM-generated summaries.

Podcast data is known to have a lot of noise mixed in (Beltagy et al., 2020), with multiple speakers, pauses, background voices, and other forms of noise that affect the quality of the final transcripts when audio data are converted to textual formats. The affected transcripts may have incorrect punctuation, wrongly interpreted or unrecognizable words due to multiple speakers and background music or voices, and some sentences might not even have an end-of-sentence punctuation mark. Textual podcast data (Shah et al., 2023) are also quite large, with our dataset having an average of 7000 tokens for each data instance, further amplifying the effects on operations performed on such data. Any text summarization performed on such data is bound to face issues when using conventional extractive summarization techniques. However, with LLMs it becomes noticeably easier, since they understand the meaning of the text and generate tokens based on what they understood from the input. They have higher capability of coping with such issues and as the size of LLMs increases, this advantage is further highlighted. Larger models also have the ability to perform more complex tasks like ignoring advertisements, generating summaries with various conditions like generating specific number of words and following certain writing patterns.

For our research, we have used podcast transcripts generated by Audio-to-Text conversion approaches (Elakkiya et al., 2022), for multiple reasons. On one hand, compared to audio, there is a larger amount of Hindi text data, so various LLMs can easily recognize pattens in textual data more effectively. Secondly, there are fewer models that recognize Hindi audio that are being used for summarization purposes (González-Gallardo et al., 2020). We also found that besides being rare, such audio recognition models show worse results when compared to their textual counterparts.

Zero-shot LLM-based summarization applied to podcast data depends on the quality of a model and finetuning, which requires a lot of resources. They might also have insufficient reasoning capabilities which may cause them to miss out on relevant details discussed in the podcast. Podcasts transcripts can also be pretty long so LLMs might forget some earlier details during processing, which also reduces the quality of the generated summary. Different LLMs compared in this study show that these traditional approaches are unable to show great results in one pass, with some models even failing to understand the instructions provided by the user (Tie et al., 2024). Our study showed that LLMs lack a clear understanding of the original text, even though the summaries were preferred by humans. Some summaries were factually incorrect, while others were too large or small even though explicitly the number of words was clearly specified in the prompts. We also found LLMs repeating previous ideas repeatedly even after introducing repetition penalties. All these show that LLMs still lag behind humans in their understanding of instructions given.

In this paper, we have come up with a framework to further optimize the current LLMs for Hindi Podcast summarization by using compressions and repeated prompts with changes to boost the factual accuracy of the final summaries, while correcting the existing issues. We also aim to reduce redundant words in the summaries through our approach, which has been a major drawback of small and medium-sized models. We have also added methods to summarize using multiple LLMs and selecting the best output, to maximize the potential of existing LLMs and our resources.

2 Related Work

Automated summarization of podcast transcripts has primarily focused on English. The TREC Podcasts tracks (2020–2021) spurred several approaches that address the unique challenges of spoken content (noisy ASR transcripts, conversational structure, very long inputs). For example, (Karlbom and Clifton, 2020) tackle the length issue by replacing BART's self-attention with Longformer's sparse attention, enabling input of thousands of tokens (Tanaka et al., 2021). (Zheng et al., 2020) propose a two-phase abstractive pipeline: they first extract important sentences from the transcript and then feed those to a pretrained encoder–decoder (e.g. BART) to generate the summary. Similarly, (Manakul and Gales, 2020) use a hierarchical filtering model to remove redundant sentences before fine-tuning a BART model (with a sequence-level reinforcement objective) on the remaining transcript.

Most existing work, however, has assumed English data. To extend summarization to other languages, (Tanaka et al., 2021) explore multilingual podcast summarization (English and Portuguese) using the Spotify dataset (Karbalaee, 2023). They fine-tune mBART-50 (a 50-language BART model) on bilingual podcast (and news) data and find that a single multilingual model performs on par with language-specific models. They also adapt mBART to a Longformer version (increasing the token limit from 512 to 4096) to better handle long transcripts (Tanaka et al., 2021) (Beltagy et al., 2020), although their Longformer variant did not outperform the base mBART in practice. These studies highlight that long-input architectures (like Longformer (Tanaka et al., 2021)) and multi-phase pipelines are crucial for podcast summarization, but they have not been evaluated on Hindi data. In short, while several systems address English podcast summarization (using Longformer attention (Beltagy et al., 2020) or multi-stage extraction-abstraction (Zheng et al., 2020)), none target Hindi episodes. This gap motivates our focus on summarizing Hindi podcast transcripts.

Compressing sentences (deleting non-essential spans) offers a middle ground between extractive and abstractive summaries. Earlier pipeline methods combined sentence selection with syntactic compression rules (e.g. ILP-based trimming of parse trees (Li et al., 2014)). More recently, neural approaches learn what spans to delete. (Desai et al., 2020) introduce a data-driven compressive model that scores each candidate deletion by two learned criteria: plausibility and salience. A deletion is plausible if it preserves grammaticality and factuality, and it is salient if it removes important information. Only spans that are plausible to delete and not highly salient are removed (Desai et al., 2020). Integrated into an extract-then-compress pipeline, this approach yields fluent, informative summaries and generalizes across domains. Such compressive models are relevant for our work because they can shorten Hindi transcripts (making them more manageable for abstractive summarizers) while maintaining coherence.

Beyond single-model baselines, recent researches aim to improve or evaluate summarizer outputs. One class of methods uses multi-stage or pipeline architectures. For example, Summ$Ñ$ (Zhang et al., 2022) is a multi-stage summarization framework for long documents: it splits a long input into chunks, generates a coarse summary in each stage, and then refines these into a final summary. This split-then-summarize strategy can handle arbitrarily long inputs with fixed-size LMs (Zhang et al., 2022). Similarly, some works iteratively refine summaries. (Wang et al., 2024) propose a summarization pipeline for user data where an LLM generates an initial summary and then applies self-critique and revision steps to reduce hallucinations and improve quality. Another direction is to use ensembling or multi-agent generation. (Fang et al., 2024) introduce a multi-LLM summarization framework, where multiple large language models collaboratively generate and evaluate summaries. They report that this multi-LLM ensemble often outperforms any single-model baseline.

Summing it all up, prior research has made progress on English podcast summarization (using long-input models and multi-phase pipelines (Karlbom and Clifton, 2020) (Singh et al., 2024)), on compressive summarization for general text (Desai et al., 2020) , and on multilingual summarization with mT5/mBART for Indian languages (Taunk and Varma, 2023) (Singh et al., 2024). However, Hindi podcast summarization remains unstudied. Our work fills this gap by integrating compressive summarization with multilingual transformer-based summarizers for Hindi audio transcripts, and by leveraging compressive summarization to enhance summary faithfulness and relevance.

3 Dataset

Our dataset of Hindi Podcast transcripts and their Podcast Descriptors was collected by crawling the web for podcast audio and their descriptors, selecting on Hindi audio for our experiments. We selected podcasts from various genres like children's stories, works on religion, astrology, and various others. The audio transcripts for the podcasts were generated using Azure Speech-to-Text services. The total number of such transcripts is 1955, along with their descriptors. These descriptors will be used as gold standard summaries for our research. Table 1 describes some of the dataset metrics for reference.

Statistic	Value
Number of Instances	1955
Total Words in Transcript	1428435
Mean Words in Instance(Transcript)	730
Max Words in Instance(Transcript)	2767
Total Words in PD	97806
Mean Words in Instance(PD)	204
Max Words in Instance(PD)	50

Table 1: Dataset Metrics

4 A Preliminary Study of Hindi Podcast Summarization Using LLMs

To study zero-shot summarization capabilities of LLMs for Hindi Podcast summarization, we selected different large language models and evaluated the summaries generated by them for our dataset on different human evaluation metrics including grammaticality, non-redundancy, referential clarity, focus, structure and coherence.

4.1 Selection of Models

For our research, we selected the following models. Later, we shall discuss ways to optimize these models for summarization of Hindi Podcasts.

- **MT5 - XLSum** (Hasan et al., 2021): This is an mT5 (Raffel et al., 2019) transformer by Google fine-tuned on the XL-Sum dataset (Hasan et al., 2021). It has approximately 580M parameters. We selected this model because it is known to have high summarization scores compared to models in its size range.

- **Gemma** (Team et al., 2024): This lightweight transformer developed by Google built using the same architecture as Gemini by Google has 2B parameters. We selected this model because despite its small size, it showed a good understanding of Hindi.

- **OpenHathi** (Mangrulkar et al., 2025): It was developed by Sarvam AI for Indian languages and has 7B parameters. This model was developed by an Indian company on a variety of Indian language texts, and despite it lacking in reasoning capabilities, gave fluent summaries.

- **GPT-3.5-Turbo** (Espejel et al., 2023): Well-known transformer developed by OpenAI. It is a state-of-the-art performance, but has limited reasoning ability. It has approximately 20B parameters.

- **GPT-4o** (Espejel et al., 2023): Another model developed by OpenAI and successor to GPT-3.5. It is larger and claims to have better and faster token generation. It has approximately 200B parameters.

4.2 Human Evaluation of Summaries

The summaries generated by our chosen models are abstractive in nature, and it is difficult to evaluate abstractive summaries with traditional methods. Hence for our study we have used the human evaluation approach to grade the models on summarization capabilities in Hindi. The summaries are annotated on five metrics to study which model performs the best. Our annotators were asked to rate the summaries on the following metrics:

- **Grammaticality**: The summary should be grammatically correct and easily readable. This includes capitalization errors, missing words, fragments, and other issues that make the summary difficult to read.

- **Non-redundancy**: There should be no unnecessary repetition of words, phrases or sentences in the summary.

- **Referential Clarity**: It should be easy to identify who or what if being referred to in the summary. There should be no objects or people who don't have a clear role in the summary.

- **Focus**: The sentences in the summary should be connected to the topic and be related to the rest of the summary.

- **Structure and Coherence**: The summary should not be a heap of unconnected but important information. It should be well-structured and well-organized, and the reader should be able to easy connect the dots on reading the summary.

4.3 Drawing a Comparison

We generated summaries for the same 50 instances from our dataset using all our selected models. We then had those summaries annotated for the metrics for human evaluation discussed before on a Likert scale of 1 to 5. The annotation was performed by 3 annotators who were MS students who were proficient in Hindi pursuing a degree in Computer Science stream. They were given random model-generated summaries and sliders for the metrics to rate. The final scores for all summaries were then averaged over the 50 summaries to get the final rating as an indicator for model performance.

As Figure 2 shows, after averaging the scores for all metrics, GPT-3.5 turbo outperforms GPT-4 in terms of overall summary quality, and GPT-4 still scores higher in terms of grammaticality. For further research, we will draw comparisons from summaries generated by GPT-3.5-Turbo.

5 AF-HPSum: Our Proposed Framework

Our framework **AF-HPSum** (Agentic Framework for Multi-Strategy Summarization of Hindi Podcasts) allows users to select different models and approaches, generate summaries and then after scoring, decide the best summary which will be provided to the user. There are multiple phases in which our research on this framework was conducted. We shall go over these phases in detail in this section.

5.1 Deletion-based Compressive Summarization Approaches

After going through the summaries generated by the given models, we still found the summaries to be lacking in terms of content, particularly in terms of coherence, readability and deviation from the transcript content. Smaller models lacked clarity and focus, reducing the overall quality of the summaries generated. To mitigate these issues, we came up with two approaches with a similar concept, which was deleting certain words from the summary without harming the integrity and meaning of the summary while reducing the overall number of words. Then, we asked the LLM to add more words to the summary. We keep repeating these steps this until there are no further deletions found by the approach, or after a fixed number of re-generations.

Algorithm 1 Compressive Summarization Pipeline

Require: Original Text, Max Loops, Threshold
Ensure: Final Summary
1: Summary $\leftarrow$ LLM(Original Text)
2: Compressed Summary $\leftarrow$ Summary
3: Loops $\leftarrow 0$
4: **while** Words(Compressive Summarization Pipeline(Compressed Summary)) - Words(Compressed Summary) $\leq$ Threshold && Loops $\leq$ Max Loops **do**
5: Compressed Summary $\leftarrow$ Compressive Summarization Pipeline(Compressed Summary)
6: Loops $\leftarrow$ Loops $+ 1$
7: **end while**
8: Final Summary $\leftarrow$ Compressed Summary

Algorithm 1 discusses the working of the compression pipeline for the approaches. We came up with to approaches that follow similar patterns to delete non-salient phrases:

- **Asking LLMs themselves to find deletions and delete them**: We can ask the LLMs to find words in the generated summaries that can be deleted, which we then delete.
- **Deriving a logic for deletions based on POS(Part-Of-Speech) tags**: Certain words or groups of words in piece of text can be deleted without affecting the final meaning of a piece of text, which we can capture and delete.

The following headings will discuss these approaches in detail.

5.2 Prompts for Compressive Summarization

Before we discuss the approaches we have introduced with our framework, we shall discuss the prompts that we will be using in conjunction with our approaches:

5.2.1 Primary Summary Prompt

We used the following prompt to ask the LLM to generate an initial summary of a piece of Hindi text:

Generate a summary of the following piece of text in Hindi Language in {number_of_words} words. Do not use any words from any other language. Be concise and informative. Do not generate any text other than the summary. The text starts from below:
{input_text}

Where:

- $target\_summary\_length$ = Words in input text multiplied by some factor (We chose 0.2 in our case)
- $input\_text$ = Text in Podcast transcript

5.2.2 Compression Prompt

The following prompt as used to ask the LLM to delete unneeded words/phrases mentioned in 5.2.5 in a piece of Hindi text summary:

I want to compress the below Hindi text by removing parts that are not required to get the complete meaning of the text. Please delete the words and phrases that can be deleted without affecting the meaning of the text. Do not generate words other than the compressed summary. The summary starts from below:
{current_summary}

Where:

- $current\_summary$ = Summary in current stage

5.2.3 Prompt to Add More Words

We used the following prompt to ask the LLM to add more words to the current summary based on a number of words deleted in the deletion step:

I have a piece of Hindi text below:
{input_text}
Could you please add {difference_of_words} more words to the following summary so that it becomes more informative and complete? Do not use any words other than Hindi language. Generate only the summary, and no other words. The summary starts from below:
{current_summary}

Where:

- $input\_text$ = Text in Podcast transcript
- $differnce\_of\_words$ = Difference of words in current and target summary length

- $current\_summary$ = Summary in current stage

5.2.4 LLM-Based Deletions

In this approach, we ask the LLMs themselves to delete the words or word groups that will not affect the final meaning of the summary. From algorithm 1, Our original summary generated using prompt 5.2.1 is fed to the LLM along with the compression prompt 5.2.2 to delete non-salient words which produces an output summary. This summary is then passed to the LLM again along with another prompt 5.2.3 to add more words to the summary depending on its current size to add more words to it. This summary is then fed back to the model again to check the threshold and further possible deletions. If we can perform further operations to it, we run operations 5.2.2 and 5.2.3 again, then check the condition mentioned in Algorithm 1 again, until we reach the threshold or run out of iterations.

5.2.5 Rule-Based Deletions

In this approach, the summary generated using prompt 5.2.1 is first parsed to obtain the POS(Part-Of-Speech) tags for its words. From these words, we selected certain words/word groups that can be deleted without affecting the meaning of the initial summary. We identified that the following can be deleted without changing the meaning of the final text:

- Adjectives/Adjectival Phrases
- Adverbs/Adverbial Phrases
- Prepositions/Prepositional Phrases
- Fragments
- Parentheticals

After deleting the mentioned word groups from the summary, we ask the LLM to add more words to it using prompt 5.2.3, completing the loop structure. After termination conditions discussed in Algorithm 1 have been met, the pipeline returns the output summary to the user.

5.3 Framework Architecture

We noticed that the summaries generated by these approaches is sometimes better than summaries with just one pass, and these chances become notably higher in the case of smaller models which struggle to understand the text in just one pass. To efficiently tackle summarization tasks, we are introducing a new framework that tackles summarization tasks more efficiently, by giving the user a choice of the models and the approaches to be used to generate the summary. Our framework, as shown in Figure 1 shall then select the best summary out of all approaches, which will be the final summary, using the model we will discuss in 7.3.

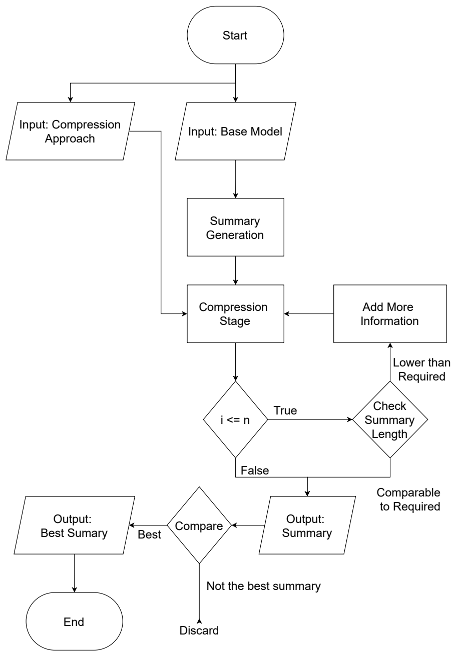

Figure 1: Framework Architecture

6 Experimental Setup

We have 2 experimental setups for our research. We shall go through the setup in detail. The following are the hardware we used for our experiments:

- For summary generation using our pipeline, we used Nvidia A100 GPUs on Google Colab. For normal pipeline runs using models that are not hosted by OpenAI like Gemma, MT5-XLSum and OpenHathi, we will use this setup
- For models like GPT-3.5-Turbo and GPT-4o, we will be using the OpenAI API for summary generation in tandem with Google Colab for our summary selection model
- For training our summary selection model, we used an Nvidia RTX 2080 Super

7 Results and Discussion

In this section we shall discuss the results for the Human Evaluation done on the base models, compare our approaches with the base architectures and discuss the results of the training of our Summary Selection model.

7.1 Single-Pass Zero-Shot Summarization

While comparing zero-shot summarization using our chosen LLMs, we noticed some interesting facts. We found that a larger number of model parameters greatly affects the summarization capability of an LLM, and also the capability to execute instructions in general. We can see from the human evaluation results that bigger models have a tendency to give a better impression, though this is not always the case which we found while comparing GPT-3.5 Turbo and GPT-4o. Although possessing a higher number of parameters, it still underperformed when compared to its predecessor because it had a tendency to exaggerate facts. This caused it to have higher grammaticality scores while lagging behind in others.

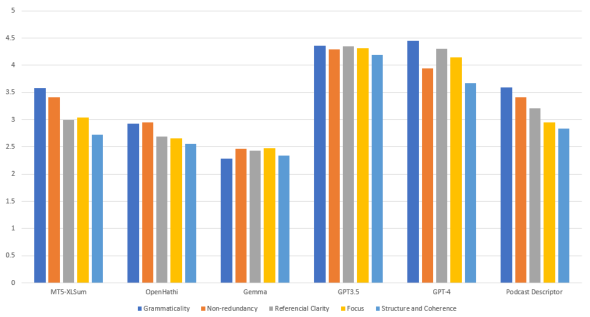

Figure 2: Human Evaluation Results for Generated Summaries

7.2 Comparison of Deletion-based Summarization Approaches

In this part, We shall go over the comparison of BERTScores for our approaches with their base models. This is different from our previous comparison of scores for Human Evaluation, the primary reasons being that we have already judged the capabilities of the base models and only need to compare them with our new approaches, so the differences might not be observable in Human Evaluation. Computed metrics like BERTScore, which check for semantic similarity shows the minute differences between the approaches better. In Table 2 we clearly see improvements over the scores of the base models in many cases. It must be kept in mind that these results reflect one run of our pipeline, since finding the mean would take a lot of time and computational power. We noticed that LLM-based deletions tend to give better scores in Precision while Logic-based deletions give better scores in Recall, in many cases.

Model	Precision	Recall	F1-Score
GPT 3.5	**0.623**	0.641	0.632
LLM Deletions GPT 3.5	0.602	**0.653**	**0.626**
Logic Deletions GPT 3.5	0.585	0.631	0.607
GPT- 4o	0.616	0.668	0.641
LLM Deletions GPT- 4o	**0.625**	**0.690**	**0.656**
Logic Deletions GPT- 4o	0.608	0.645	0.626
Openhathi	0.605	0.634	**0.619**
LLM Deletions Openhathi	**0.624**	0.577	0.599
Logic Deletions Openhathi	0.556	**0.645**	0.597
Gemma - 2B	0.612	**0.653**	**0.6323**
LLM Deletions Gemma - 2B	**0.638**	0.589	0.612
Logic Deletions Gemma - 2B	0.628	0.634	**0.631**

Table 2: BERTScore Comparison vs Podcast Descriptor

7.3 Selection of Final Summary

After the generation of summaries from our various approaches has concluded, the framework will proceed with the selection of the best summary. For this, we have trained a model on generated summaries for our dataset. We used various LLMs for generating the embeddings of our summaries, then finally selected IndicBERTv2-SS (278M parameters) (Doddapaneni et al., 2023) as the final model, since it gave vastly better results for our dataset during our comparisons. The embeddings generated by this model will be concatenated and be used as inputs for our model. The outputs for training the model will be decided by the summary scores of the individual models, where the output best summary will be set to 1 and the rest will be set to 0. The summary scores are decided by a weighted average of BERTScore, BleuScore and ROUGE-Scores, with their contribution to the final scores being 50%, 25% and 25%. Currently, **AF-HPSum** supports only one such model with 4 input summaries, but we plan to introduce more such models later with 5 or more model-approach combinations to further enhance our framework. We allocated 768 tokens for each summary so the number of input neurons for this model are 768×4 and the number of outputs are 4. There are 2 hidden layers with 1024 and 512 neurons respectively. We used gumble-softmax as the activation function for the output layer.

Figure 3 show the results of tuning the hyperparameters of the model with 4 input summaries and their scores. The final results of the tuning showed an mean accuracy of 87.18% and loss of 0.3347 after testing the best model.

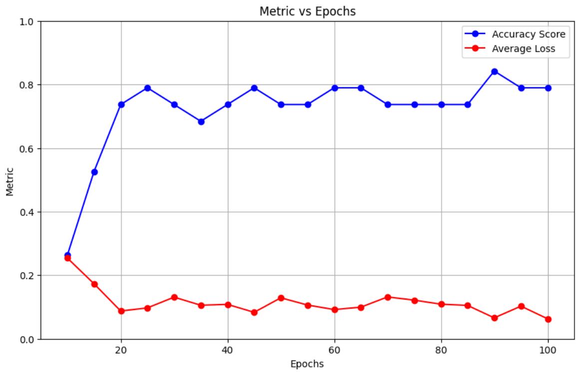

Figure 3: Hyperparameter Tuning for Selection Model - Accuracy and Loss vs Epochs

8 Conclusion and Future Work

Our research shows that there are still ways to optimize summarization, even with advanced LLMs, since they might miss key points which may be overlooked during automation. Prompt engineering can help in most cases, but for regional languages where the data is scarcer, LLMs may still fail to produce desirable results. Hence, they still require experts to fix issues to get the maximum benefit. To mitigate these issues we propose a framework for the generation of Podcast summaries, with options to select the model and multiple approaches that can further optimize the quality of the summaries. We also proposed two approaches to increase summary focus using deletion based summarization which proved to be better for smaller models.

We found it better to use multiple LLMs and then comparing the summaries using smaller models to judge their quality to get the best results. In future, we can enhance our framework by introducing more approaches, support for more models, and even languages.

9 Limitations

In this section, we would like to clarify some limitations associated with our paper.

- **Computational Power**: Even though we aim to enhance the quality of summaries of smaller models, some degree of computational power is required to inference using the models.
- **Time**: The time to reach the output stage is way higher than if we just use the base models.
- **Dataset**: We have only tested our models for one language, i.e. Hindi, which we later plan to rectify.
- **Summary Selection Model**: Currently we only have one summary selection model, which can accommodate at most 4 model-approach selections.
- **Advancements in Base Models**: Our current approaches may degrade in a cost-performance analysis with more advanced models.
- **Text Summarization**: We are not summarizing Podcast audios directly. A medium is necessary to transcribe the audio.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Shrey Desai, Jiacheng Xu, and Greg Durrett. 2020. Compressive summarization with plausibility and salience modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6259–6274, Online. Association for Computational Linguistics.

Sumanth Doddapaneni, Rahul Aralikatte, Gowtham Ramesh, Shreya Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. 2023. Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12402–12426, Toronto, Canada. Association for Computational Linguistics.

A. Elakkiya, K. Jaya Surya, Konduru Venkatesh, and S. Aakash. 2022. Implementation of speech to text conversion using hidden markov model. In *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, pages 359–363.

Jessica López Espejel, El Hassane Ettifouri, Mahaman Sanoussi Yahaya Alassan, El Mehdi Chouham, and Walid Dahhane. 2023. Gpt-3.5, gpt-4, or bard? evaluating llms reasoning ability in zero-shot setting and performance boosting through prompts. *Preprint*, arXiv:2305.12477.

Jiangnan Fang, Cheng-Tse Liu, Jieun Kim, Yash Bhedaru, Ethan Liu, Nikhil Singh, Nedim Lipka, Puneet Mathur, Nesreen K Ahmed, Franck Dernoncourt, and 1 others. 2024. Multi-llm text summarization. *arXiv preprint arXiv:2412.15487*.

Carlos-Emiliano González-Gallardo, Romain Deveaud, Eric SanJuan, and Juan-Manuel Torres. 2020. Audio summarization with audio features and probability distribution divergence. *CoRR*, abs/2001.07098.

Tahmid Hasan, Abhik Bhattacharjee, Md Saiful Islam, Kazi Samin, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. Xl-sum: Large-scale multilingual abstractive summarization for 44 languages. *CoRR*, abs/2106.13822.

Mohammad Karbalaee. 2023. Analysis on the spotify dataset.

Hannes Karlbom and Ann Clifton. 2020. Abstractive podcast summarization using bart with longformer attention. In *The 29th Text Retrieval Conference (TREC) notebook. NIST*.

Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. Improving multi-documents summarization by sentence compression based on expanded constituent parse trees. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 691–701.

Potsawee Manakul and Mark Gales. 2020. Cued_speech at trec 2020 podcast summarisation track. *arXiv preprint arXiv:2012.02535*.

Sourab Mangrulkar, Dhanush Reddy, and 1 others. 2025. OpenHathi-Instruct: Instruction-tuned Bilingual OpenHathi for Hindi and Hinglish. GitHub repository. Available at `https://github.com/pacman100/openhathi_instruct`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Neil Shah, Vivek Srivastava, Mohit Bhardwaj, Satej Kadlay, Dharmeshkumar Agrawal, Savita Bhat, and Niranjan Pedanekar. 2023. It's what you say and how you say it: Exploring audio and textual features for podcast data. In *2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1972–1977.

Geetanjali Singh, Namita Mittal, and Satyendra Singh Chouhan. 2024. Hindisumm: A hindi abstractive summarization benchmark dataset. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 23(12):1–15.

Edgar Tanaka, Ann Clifton, and Md Iftekhar Tanveer. 2021. Multilingual podcast summarization using longformers. In *TREC*.

Dhaval Taunk and Vasudeva Varma. 2023. Summarizing indian languages using multilingual transformers based models. *arXiv preprint arXiv:2303.16657*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, and 89 others. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Jiessie Tie, Bingsheng Yao, Tianshi Li, Syed Ishtiaque Ahmed, Dakuo Wang, and Shurui Zhou. 2024. Llms are imperfect, then what? an empirical study on llm failures in software engineering. *arXiv preprint arXiv:2411.09916*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Chao Wang, Neo Wu, Lin Ning, Jiaxing Wu, Luyang Liu, Jun Xie, Shawn O'Banion, and Bradley Green. 2024. Usersumbench: A benchmark framework for evaluating user summarization approaches. *arXiv preprint arXiv:2408.16966*.

Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed Awadallah, Dragomir Radev, and Rui Zhang. 2022. Summ^n: A multi-stage summarization framework for long input dialogues and documents. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1592–1604, Dublin, Ireland. Association for Computational Linguistics.

Chujie Zheng, Kunpeng Zhang, Harry Jiannan Wang, and Ling Fan. 2020. A two-phase approach for abstractive podcast summarization. *arXiv preprint arXiv:2011.08291*.