
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2023 SCPE. Volume 24, Issues 4, pp. 835–846, DOI 10.12694/scpe.v24i4.2248

SENTIMENT ANALYSIS AND SPEAKER DIARIZATION IN HINDI AND MARATHI USING USING FINETUNED WHISPER

GOWTHAM DORA PAPPALA, ANIKET RANSING, AND POOJA JAIN*

Abstract. Automatic Speech Recognition (ASR) is a crucial technology that enables machines to automatically recognize human voices based on audio signals. In recent years, there has been a rigorous growth in the development of ASR models with the emergence of new techniques and algorithms. One such model is the Whisper ASR model developed by OpenAI, which is based on a Transformer encoder-decoder architecture and can handle multiple tasks such as language identification, transcription, and translation. However, there are still limitations to the Whisper ASR model, such as speaker diarization, summarization, emotion detection, and performance with Indian regional languages like Hindi, Marathi and others. This research paper aims to enhance the performance of the Whisper ASR model by adding additional components or features such as speaker diarization, text summarization, emotion detection, text generation and question answering. Additionally, we aim to improve its performance in Indian regional languages by training the model on common voice 11 dataset from huggingface. The research findings have the potential to contribute to the development of more accurate and reliable ASR models, which could improve human-machine communication in various applications.

Key words: Automatic Speech Recognition, Whisper, Summarization, Diarization, Question Anwering, Emotion Detection, Text Generation

1. Introduction. Automatic Speech Recognition (ASR) is the technology that allows human beings to speak with a machine or a computer interface. The increasing prevalence of speech-to-text systems has revolutionized the way people interact with their devices by allowing them to use voice commands. However, most of these systems are trained on datasets from major languages and do not perform well on Indian languages. Therefore, creating accurate automatic speech recognition (ASR) systems for Indian languages has become a critical research issue. This study aims to develop a highly effective ASR system for Indian languages that can achieve an impressively low word error rate (WER). To achieve this objective, we are fine-tuning the Whisper pre-trained model using Indian datasets and incorporating additional features to enhance the system's performance. The added features include diarization, summarization, translation, and question answering capabilities. The aim of this project is to build a perfect ASR which can be used at College level or local level at least. This ASR system can help students as well as the administrative authority to carry out transcription or translation activities in any department and students can use this system in their mini projects if necessary. We want to build a proper multilingual ASR which can work over regional languages of India which has been our major target. This paper outlines our methodology for creating the ASR system, including the datasets used, fine-tuning techniques, and the added features. We then evaluate the system's performance and compare it to existing ASR systems. Finally, we discuss the implications of our research and suggest future research avenues.

2. Literature Review. Various Projects and Papers have been proposed regarding Automatic Speech Recognition in the past. Most of the work done in this field is regarding finetuning a pretrained model on huge amounts of labeled data and concentrating only on speech to text transcription which is considered as core part of the Speech Recognition. Here are some works of the ASR. As mentioned in the paper [2], the model can perform Language Identification,Speech to Text Transcription, and Language Translation simultaneously making Whisper an extraordinary model. In the paper [1] the authors explain about Zero Shot Classification which means that classifying objects though it has never trained on them, that means model being able to generalize well on other unknown data where it has never been trained. In the paper [4], the authors explain how finetuning any pre trained model with any model on any specific data works well with that data compared

*Indian Institute of Information Technology, Nagpur

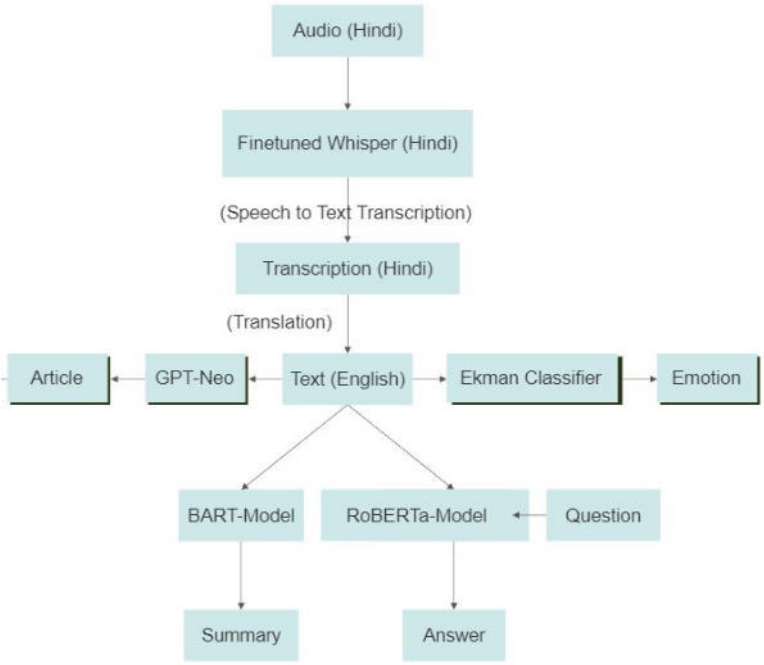

Fig. 3.1: Work Flow

to previous performance without any finetuning. In the paper [3], the Wav2Vec2.0 pretrained model which is trained on so many hours of unlabeled data is finetuned with transformer model on LibriSpeech dataset and achieved State of the Art. In the paper [5], the authors describe the architecture and pretraining objectives used in BART, which is based on the same architecture as the GPT models but uses denoising autoencoding as a pretraining objective. In the paper [7], it is a variant of the BERT model that uses a larger training corpus and a longer pretraining schedule. They fine-tune RoBERTa on the SQuAD 1.1 and 2.0 datasets, achieving state-of-the-art performance on both for question answering tasks. The paper [6] builds on the success of d-vector based speaker verification systems to develop a new d-vector based approach to speaker diarization by combining LSTM-based d-vector audio embeddings with recent work in nonparametric clustering to obtain a state-of-the-art speaker diarization system. In the paper [9] the authors propose a parameter-efficient approach for fine-tuning large pre-trained transformer models. In the paper [8], the GPT-Neo model is a series of transformer-based language models that are trained on a massive amount of text data. It is similar in architecture to the original GPT models, but uses a different training procedure and can scale to much larger model sizes. The paper describes the architecture and training procedure for GPT-Neo, as well as the results of experiments on a range of natural language processing tasks, including text generation, where it achieves state-of-the-art results on several benchmarks.

3. Work Done. In this section we present the work done based on the literature review and our own ideas. We have shown the design of our system and the various tools and frameworks adopted in the process.

3.1. Introduction to Whisper. Whisper is an automatic speech recognition (ASR) system that has been trained on a vast amount of multilingual and multitask supervised data. The model, which was published in September 2022 by Alec Radford and his team at OpenAI, is pre-trained on a massive amount of labeled audio-transcription data - precisely 680,000 hours. This sets it apart from its predecessors, such as Wav2Vec2.0, which are pre-trained on unlabelled audio data. Thanks to its extensive pre-training, Whisper has various model checkpoints that can be applied to over 96 different languages. The model has demonstrated a remarkable ability to generalize to many datasets and domains, achieving competitive results to state-of-the-art ASR sys- tems.

On the test-clean subset of LibriSpeech ASR, Whisper achieved a word error rate (WER) of around 3 percent. It also set a new state-of-the-art record on TED-LIUM with a WER of 4.7 percent. The multilingual ASR knowledge acquired during pre-training can be leveraged for other low-resource languages through fine-tuning. The pre-trained checkpoints can be adapted for specific datasets and languages to further improve upon these results. In particular, our team has focused on fine-tuning the Whisper model on Indian datasets to achieve state-of-the-art results in the language we are fine-tuning.

3.2. Why We Choose Whisper. There are several reasons for us to choose whisper apart from its superior performance. The Whisper model is open source and made public by OpenAi due to which we are able to now create an ASR for Indian Languages with this pretrained model and it can also do other tasks apart from speech to text transcription, such as Language Identification, Language Translation and Voice Activity Detection which makes it suitable for Multitask training and moreover all these features are integrated in a single pipeline which makes this model even extraordinary. The whisper model is Multilingual which means it can perform all its features on 96 other languages along with English as it was trained on some vast amount of data which includes both English and 96 other languages. Moreover the authors state that the Whisper model achieves human level performance in English speech recognition. So we took this as a challenge and wanted to integrate the Whisper model which can perform well on our Indian languages.

3.3. How Whisper Benefits us:. We will be able to transcribe large audio files, such as podcasts. Here we are looking to transcribe Hindi lecture audios or podcasts with very less word error rate and then further summarize the transcript so that the reader can know the context of the lecture within a short time. We can also create accurate subtitles for our Youtube videos or other content. Also, using a non-English language is not a limitation. Whisper has a feature of translation which makes it more beneficial. One of the most likable factors which can benefit the most is that People with hearing impairment will have a much better quality of life. And also we are not stopping with the features that are available within whisper, we also got an edge to add additional components or features like summarization, text emotion detection, text generation, diarization and question answering.

3.4. Architecture of Whisper. It is implemented as encoder-decoder architecture. It takes in 30-second chunks of audio input and converts into a logMel spectrogram. This spectrogram is now processed by a two layered CNN with GELU activation functions and further enriched with sinusoidal position embeddings. The input is now given to the Encoder part of the Transformer for processing. Decoder predicts corresponding text captions. Special tokens have been added which helps in performing further tasks like language identification, phrase-level timestamps, multilingual speech transcription, and to-English speech translation.

3.5. Multitasking of Whisper. Speech Recognition has a limitation on predicting which words were spoken in a given audio snippet and there has been a lot of research on that. A fully featured speech recognition system can involve many additional components or features like voice activity detection, speaker diarization, language identification and language translation. These components are often handled separately, resulting in a complex system around the core speech recognition model. To reduce this complexity, whisper model have been designed in such a way that it performs the entire speech processing in a single pipeline. The different tasks that can be performed by the whisper on the same input audio signal are Language Identification, Language Translation and Voice Activity Detection.

3.6. Whisper Configurations. As previously mentioned, the Whisper model has been trained on a vast amount of multilingual and English data, allowing it to support more than 96 languages. Whisper is available in five configurations, each with different parameters and layers. Four of these configurations were trained exclusively on English data, while all five were trained on multilingual data. Consequently, all configurations are capable of working with multilingual data, but their performance may vary. The size of a model impacts its performance, and models with more parameters and layers tend to perform better. A table is provided below that details each model configuration's width, the number of layers it contains, and the number of parameters it contains.

3.7. Whisper Finetuning. We demonstrated how the pre-trained Whisper checkpoints can be fine-tuned on any multilingual ASR dataset. We installed several popular Python packages to fine-tune the Whisper model.

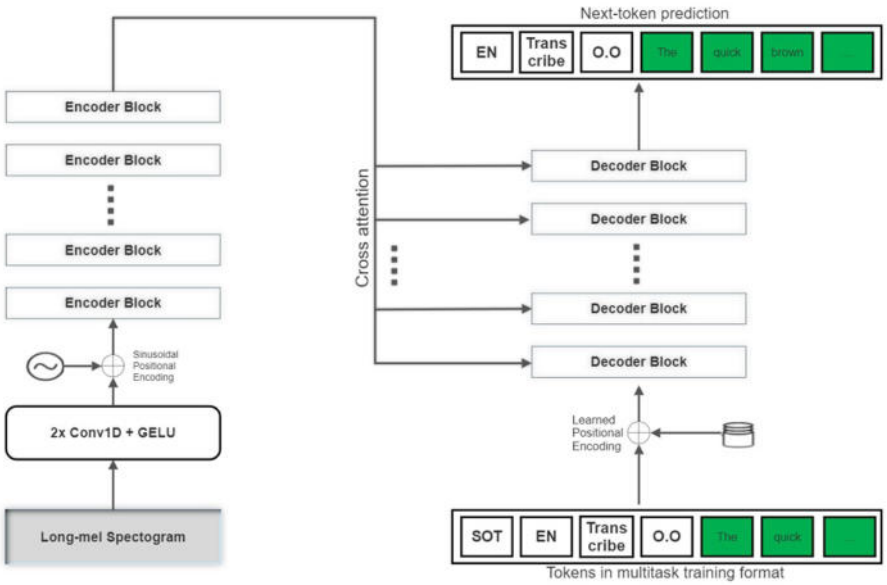

Fig. 3.2: Architecture of Whisper

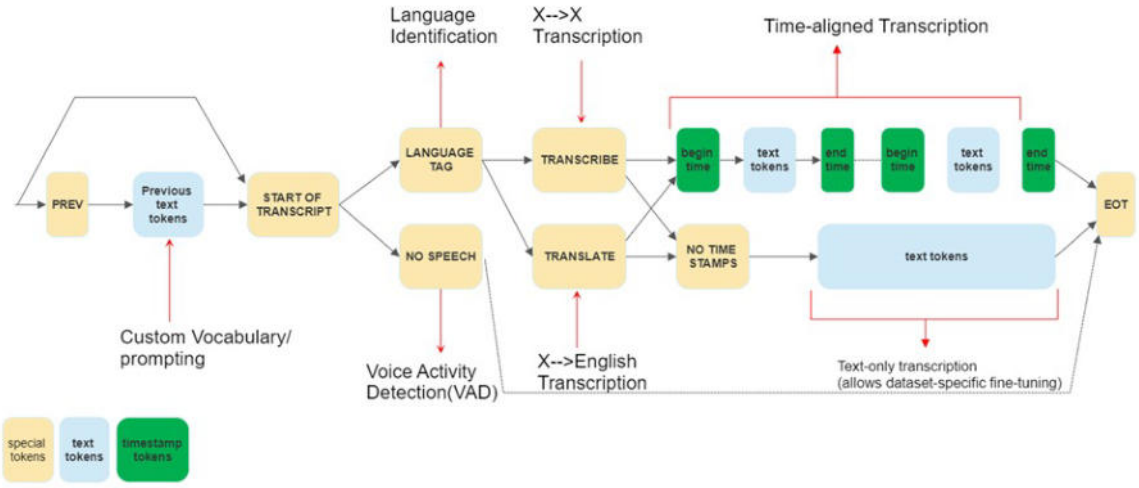

Fig. 3.3: Multitask training format

We downloaded the dataset and prepared our training data using the dataset package and we loaded and trained our Whisper model using huggingface transformers. We used soundfile package to pre-process audio files and evaluate, and used jiwer to assess the performance of our model. We used the Common Voice version 11 dataset available from huggingface for training and validation. we fine-tuned our model on Hindi, Marathi and Tamil languages available in common voice version 11 dataset. The ASR pipeline can be divided into three components: A feature extractor which pre-processes the raw audio-inputs, and the model which performs the sequence-to-sequence mapping, and a tokenizer which post-processes the model outputs to text format. In Transformers, the Whisper model has an associated feature extractor and tokenizer, called Whisper Feature Extractor and Whisper Tokenizer respectively. For Training and Evaluation, we created a data collector which

Table 3.1: Whisper Configurations

Size	Layers	Width	Heads	Parameters	English-only	Multilingual
tiny	4	384	6	39M	Yes	Yes
base	6	512	8	74M	Yes	Yes
small	12	768	12	244M	Yes	Yes
medium	24	1024	16	769M	Yes	Yes
large	32	1280	20	1550M	No	Yes

takes our preprocessed data and prepares pytorch tensors ready for the model. During evaluation, we used the Word Error Rate (WER) metric to evaluate the model. We loaded a pre-trained checkpoint and configured it correctly for training by defining right set of parameters. After fine-tuning the model, we evaluated it on the test data. We achieved a Word Error Rate of 55.9% with the tiny model on hindi language where it took 5 hours for training and achieved a Word Error Rate of 33.7% with the small model on hindi language where it took 9 hours for training. The results could likely be improved by optimizing the training hyperparameters, such as learning rate and dropout, or by using a larger pre-trained checkpoint like either medium or large. But here we are having a limitation of using medium and large model configurations due to GPU and memory limits that we have on colab. We are able to finetune only a few model configurations on local colab and also it is consuming a lot of time in execution , so we have done research on how to use GPU or limited memory efficiently. The training parameters used for finetuning the whisper model were shown below.

3.8. Parameter Efficient Finetuning with Whisper. Parameter Efficient Fine-tuning is a technique used in machine learning to adapt a pre-trained model to a new task with limited training data and computational resources. We have followed the same steps as previous finetuning approach but here we used Parameter Efficient Fine-tuning and bitsandbytes to train the whisper-large-v2 seamlessly on a colab with T4 GPU (16 GB VRAM). The idea is to modify the existing pre-trained model's parameters in a way that makes it better suited for the new task, without having to retrain the entire model from scratch. Now we are able to use large models with more weights and also can reduce the computation time. Here we will present how we used PEFT LoRA+BNB INT8 to train our model. Right after loading the Whisper large model checkpoint we applied some post-processing on the 8- bit model to enable training and freezed all our layers, and casted the layer-norm in float32 for stability. We also casted the output of the last layer in float32 for the same reason. After that we loaded a PeftModel and specified that we are going to use low-rank adapters (LoRA) using get-peft-model utility function from peft. We only used 0.67% of the total trainable parameters, thereby performing Parameter-Efficient Fine-Tuning. In the final step, we defined all the parameters related to training. PEFT is a useful technique for adapting pre-trained models to new tasks in a resource-efficient way. After training the whisper model for hindi data, we got model checkpoints with their respective training and validation loss (see Result section). After Finetuning Whisper-large-V2 for Hindi language and tested on test data we got the word error rate of 25.8% and normalized word error rate around 13% (see Result section). After training the whisper model for Marathi data, we got model checkpoints with their respective training and validation loss (see Result section). After Finetuning Whisper-large-V2 for Marathi language we got the word error rate of around 40% (see Result section). Note: We have an improved result with the whisper finetuned models that got trained on common voice hindi data with the help of PEFT and achieved an word error rate of 25 percent

```
from transformers import Seq2SeqTrainingArguments

training_args = Seq2SeqTrainingArguments(
    output_dir="./whisper-small-hi",
    per_device_train_batch_size=8,
    gradient_accumulation_steps=1,
    learning_rate=1e-5,
    warmup_steps=500,
    max_steps=4000,
    gradient_checkpointing=True,
    fp16=True,
    evaluation_strategy="steps",
    per_device_eval_batch_size=8,
    predict_with_generate=True,
    generation_max_length=225,
    save_steps=1000,
    eval_steps=1000,
    logging_steps=25,
    report_to=["tensorboard"],
    load_best_model_at_end=True,
    metric_for_best_model="wer",
    greater_is_better=False,
    push_to_hub=False,
)
```

Fig. 3.4: Training Parameters - 1

and the training got completed within six hours. So we are able to increase the performance of the model as well decrease the training time. The training parameters used for fine-tuning the whisper model were shown below.

3.9. Speaker Diarization. Diarization is a process of clustering audio or speech data into homogeneous segments based on speaker identity. In other words, it is the process of determining "who spoke when" in an audio. Diarization is one of the essential components in several applications such as speaker recognition, speech-to-text transcription, and language Identification. It involves a series of steps such as feature extraction, clustering, and classification. The first step in diarization is feature extraction, which involves transforming the audio data into a set of numerical features that can be used to distinguish between different speakers. This is usually done using techniques such as Mel-Frequency Cepstral Coefficients (MFCCs) or Perceptual Linear Prediction (PLP). Next, the extracted features are clustered to group together segments of audio that are likely to have been spoken by the same speaker. This is typically done using unsupervised clustering techniques such as K-means or Gaussian Mixture Models (GMMs). While diarization has made significant progress in recent years, it still poses several challenges. One of the main challenges is dealing with overlapping speech, where multiple speakers are talking simultaneously. Another challenge is dealing with variability in speech patterns caused by factors such as age, gender, and accent. So here we want to add this important feature in our ASR using Pyannote from Hugging Face which has a Pyannote.audio package which is an open-source toolkit written in Python for speaker diarization and has better results. So we just stitched the whisper model to this pyannote.audio for speaker diarization and the clustering algorithm we used is AgglomerativeClustering. It is an unsupervised learning technique that groups similar data points into clusters based on their pairwise distances or similarities. Agglomerative clustering also has some limitations and one of the main limitation is that the algorithm is sensitive to noise and outliers, which can disrupt the clustering process but we can ensure that this diarization task can work well on small audio clips of two English speakers.

```
from transformers import Seq2SeqTrainingArguments

training_args = Seq2SeqTrainingArguments(
    output_dir="whisper-large-v2-hindi",
    per_device_train_batch_size=8,
    gradient_accumulation_steps=1,
    learning_rate=1e-3,
    warmup_steps=50,
    max_steps=1500,
    evaluation_strategy="steps",
    fp16=True,
    per_device_eval_batch_size=8,
    generation_max_length=128,
    eval_steps=500,
    logging_steps=25,
    remove_unused_columns=False,
    label_names=["labels"],
)
```

Fig. 3.5: Training Parameters -2

3.10. Summarization. Summarization refers to the task of creating a brief and coherent summary of a longer piece of text. It is an important task in natural language processing (nlp), as it can help anyone quickly understand the key points and main ideas of a document or a longer text without having to read it completely or entirely. We have done a lot of research on Summarization for Indian Languages and it is very challenging as India is a multilingual country with a diverse range of languages and dialects. Summarization models developed for English may not be directly applicable to Indian languages due to differences in grammar, syntax, and vocabulary. There has been significant research in recent years on developing summarization models for Indian languages, including Hindi, Bengali, Tamil, Telugu, and others. We have explored both extractive and abstractive summarization approaches for Indian languages. It is not easy to develop Summarization models for Indian languages due to lack of annotated data. There are fewer annotated datasets available for Indian languages compared to English, which makes it difficult to train and evaluate summarization models. The main challenge is the need to handle the rich morphology and complex sentence structures of Indian languages. Indian languages have a rich morphology, with words often having multiple forms depending on context and usage. Sentence structures can also be complex, with long sentences and nested clauses. Summarization models for Indian languages need to take these factors into account to generate accurate and readable summaries. We have done research on the IndicNLP Library which provides many open source models specifically tasked for Indian languages. We used the Summarization model from the IndicNLP Library and it still needs some improvement though it is working pretty well on some language texts. Our main idea is to translate the transcript to english that gets transcribed from the whipser model which will be in hindi langauge considering that we are using finetuned whisper model trained on hindi language and then summarize the translated text as the state-of-the-art summarization models work pretty well on english language and also so that those who cannot understand regional languages can also know or understand the summary of a regional lecture audio or any other large audio. But still we are trying to make a summarizer model for Indian languages and in future we are going to fine-tune Pegasus from Google , summarization model from hugging face ,T5 and BertSum models on any Indian Annotated Dataset and will finish this soon. As of now we used the BART Summarization model for summarizing our transcript as it is able to derive better and meaningful summaries. BART (Bidirectional and Auto Regressive Transformer) is a state-of-the-art language model developed by Facebook AI [5]. It is a pre-trained transformer based neural network that is trained on a huge corpus of text data using a combination of unsupervised and supervised learning techniques. This model works by encoding the input text into a series

Table 4.1: Testing Whisper on LibriSpeech

Model	Parameters	WER
Tiny	39M	5.65%
Base	74M	4.27%
Small	244M	3.06%
Medium	769M	3.02%

of numerical representations using a transformer-based neural network. It then decodes these representations to generate a summary of the input text. The reason for this model to generate coherent and meaningful summaries is its bidirectional feature which means it can take both the preceding and following context when generating a summary. This makes BART's summarization model particularly effective at producing accurate and coherent summaries.

3.11. Question Answering. The goal of QA bot is to enable users to ask questions in natural language and receive accurate and relevant answers in real-time. However we are generating context using our ASR , So we built a QA bot which takes this context and also a question from the user and generates an answer. Here, the question should be related to the context, then only our model can generate correct and meaningful answers. We used a pretrained Roberta model from hugging face for our Question Answering task [7]. This model has been fine-tuned on the Stanford Question Answering Dataset (SQuAD) 2.0. SQuAD is a benchmark dataset for machine comprehension tasks, where the goal is to answer a question based on a given context passage. This model has been fine-tuned to specifically answer questions based on the SQuAD 2.0 dataset, which includes more challenging questions compared to the original SQuAD dataset. The model has been trained to identify the answer span within the given context passage that best answers the question. In our case it takes the context that was generated from ASR and a question from the user as a set of inputs and generates the answer.

3.12. Emotion Detection. Emotion detection with text is a natural language processing technique that involves automatically identifying the emotional tone or sentiment expressed in a piece of text. The goal of emotion detection is to classify a piece of text as positive or negative based on the emotional content of the text. But here we have done further research on detecting various other emotions such as Joy, Sadness, Anger, fear, surprise, disgust as specified by Paul Ekman. If not any of these it will be neutral. We found an open source model which does this type of emotion detection well and the model we used was the arphangoshal/Ekman classifier from huggingface where it takes a piece of text as input and predicts the emotion of a speaker based on the text itself. This text input is nothing but the translated English text of a transcript that we get from the Whisper model or the summarized text that we generate from the summarizer model.

3.13. Text Generation. Text generation is a process of generating new, coherent text based on a given prompt or context. Text generation is a challenging task in natural language processing, as it requires the model to generate text that is not only grammatically correct but also semantically meaningful and coherent. We used the GPT-NEO model for our text generation [8] and we generated texts or blogs using aitextgen package. There are many predefined functions available in aitextgen which are so useful for text generation tasks. So anyone who wants to generate a blog article based on any context they want or who wants to generate some contextual text based on any prompt, they can just give their audio or speak to the ASR and the transcript generated from that is given to the text generation model.

4. Experiments. We have tested Whisper on different datasets and checked the word error rates of every model configuration with some datasets as shown in this section (Tables 4.1-4.4).

Table 4.2: Comparison with wav2vec2.0

Model	Dataset	WER
Whisper	LibriSpeech	2.7%
Wav2Vec2.0	LibriSpeech	2.7%

Table 4.3: Testing Whisper on CLSRIL-23 Dataset

Model	Latency	Accuracy	Precision	Recall	F1-Score
Wav2Vec	8.97%	49.2%	38.0%	100%	55.1%
Tiny	4.98%	82.2%	86.9%	90.9%	88.8%
Base	11.14%	82.2%	84.4%	93.2%	88.5%
Small	16.94%	81.6%	80.7%	96.1%	87.7%
Medium	49.54%	90.8%	92.0%	96.5%	94.2%

Table 4.4: Testing Whisper on Translation Part on Fleurs dataset

Language used	Word Error Rate(WER)
Hindi	28.34%
Marathi	66.56%
Tamil	33.63%
Malayalam	44.60%
Telugu	84.20%

Table 5.1: Comparision between Normal Finetuned model and Finetuned model using Parameter-Efficient-Fine-tuning

Model Configuration	Finetuned Whisper WER	PEFT Whisper WER
Tiny	55.6%	64.9%
Small	33.5%	38.6%

5. Results. The word error rate of our model is 25% for hindi languae and 40% for marathi language. As mentioned in the previous section we used transformers from huggingface to finetune Whisper model on common voice 11.0 dataset. We have also shown and compared the results with other model configurations in this section. We have also shown the comparisions of fietuned whisper WER with the normal Whisper WER for every model checkpoint in this section and also shown the difference between WER of Whisper finetuned model and the model which was trained with PEFT.

[4000/4000

Step	Training Loss	Validation Loss	Wer
1000	0.347800	0.561929	64.335901
2000	0.241300	0.510025	58.659951
3000	0.191000	0.502138	56.268518
4000	0.160100	0.503013	55.989164

Fig. 5.1: Finetuned Whisper-tiny results

```
0it [00:00, ?it/s]
Reading metadata...: 0it [00:00, ?it/s]
Reading metadata...: 2894it [00:00, 16645.43it/s]
/usr/local/lib/python3.8/dist-packages/bitsandbytes/autograd/_functions.py:298
  warnings.warn(f"MatMul8bitLt: inputs will be cast from {A.dtype} to float16 
362it [1:11:33, 11.86s/it]wer=38.618471175823245
```

Fig. 5.2: Finetuned whisper-small WER

Step	Training Loss	Validation Loss
500	0.160800	0.254170
1000	0.089800	0.220722
1500	0.031900	0.211729

Reading metadata...: 2894it [00:02, 1276.94it/s]

Fig. 5.3: Finetuned Whisper-large-V2 using PEFT-Lora + BNB INT8 training + Streaming dataset for Hindi

Step	Training Loss	Validation Loss
500	0.172900	0.272142
1000	0.040800	0.290398
1500	0.010300	0.307799

Fig. 5.4: Finetuned Whisper-large-V2 using PEFT-Lora + BNB INT8 training + Streaming dataset for Marathi

```
0it [00:00, ?it/s]
Reading metadata...: 0it [00:00, ?it/s]
Reading metadata...: 2894it [00:00, 10561.80it/s]
/usr/local/lib/python3.8/dist-packages/bitsandbyt
  warnings.warn(f"MatMul8bitLt: inputs will be ca
362it [2:10:50, 21.69s/it]wer=25.895200203166002
```

Fig. 5.5: WER of Finetuned Whisper large-v2 on Hindi

6. Conclusion. Automatic Speech Recognition (ASR) technology has become an essential tool in human-machine communication. The Whisper ASR model developed by OpenAI is a promising development in this

```
0it [00:00, ?it/s]
Reading metadata...: 0it [00:00, ?it/s]
Reading metadata...: 2894it [00:00, 10456.52it/s]
362it [2:11:43, 21.83s/it]normalized_wer=13.10319567805146
```

Fig. 5.6: Normalized WER of Finetuned Whisper large-v2 on Hindi

```
0it [00:00, ?it/s]
Reading metadata...: 0it [00:00, ?it/s]
Reading metadata...: 1816it [00:00, 12578.29it/s]
/usr/local/lib/python3.8/dist-packages/bitsandbytes/autograd/_func
  warnings.warn(f"MatMul8bitLt: inputs will be cast from {A.dtype}
227it [2:11:27, 34.75s/it]wer=40.19727935013861
```

Fig. 5.7: WER of Finetuned Whisper large-v2 on Marathi

Table 5.2: Comparison between WER of Finetuned model and WER of normal model

Model Configuration	Finetuned Whisper WER	Normal Whisper WER
Small	38.6%	87.4%
Medium	29.2%	54.6%
Large-V2	25.8%	51.5%

field, with its Transformer encoder-decoder architecture capable of handling various tasks. However, the model's limitations in speaker diarization, summarization, and emotion detection, as well as its performance with Indian regional languages, need to be addressed. In this research we enhanced the Whisper ASR model's capabilities by adding features such as speaker diarization, text summarization, emotion detection, and a Question Answering. Additionally, the model's performance in Indian regional languages was improved through training on Indian languages. By addressing these limitations and improving the model's performance, this research has the potential to contribute to the development of more accurate and reliable ASR models, ultimately improving human-machine communication in various applications. Overall, this research could have a significant impact on the advancement of ASR technology and its applications in various industries. In the future this project can be extended to integrate with different apps and websites where the user would interact with our ASR and this project has a scope of improvement through increasing the training data and also by using different models.

REFERENCES

[1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever, *Learning transferable visual models from natural language supervision*, in 2018 Chinese Automation Congress (CAC), IEEE, 2021, pp. 2446–2450.

[2] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, Ilya Sutskever, *Robust speech recognition via large-scale weak supervision*, IEEE Access, 7 (2022), pp. 38044–38054.

[3] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, Michael Auli, *wav2vec 2.0: A framework for self-supervised learning of speech representations*, Journal homepage: www. ijrpr. com ISSN, 2582 (2020), p. 7421.

[4] Heng-Jui Chang, Alexander H. Liu, Hung-yi Lee, Lin-shan Lee, *End-to-end whisperd speech recognition with frequency-weighted approaches abd pseudo whisper pre-training*, in 2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE), IEEE, 2020, pp. 402–406.

[5] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer, *Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, in 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), IEEE, 2020, pp. 324–329.

[6] Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, Ignacio Lopez Moreno, *Speaker diarization with*

lstm, in 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), IEEE, 2019, pp. 324–329.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov, *Roberta: A robustly optimized bert pretraining approach*, in 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), IEEE, 2019, pp. 324–329.

[8] Sid Black,Leo Gao, Phil Wang, Connor Leahy, Stella Rose Biderman, *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow*, https://doi.org/10.5281/zenodo.5297715, 2021.

[9] Liu Liu, Zheng Qu, Zhaodong Chen, Yufei Ding, Yuan Xie, *Training Deeper Transformers with Sparse Attention via Conditional Computation*, arXiv:2110.11299v1, 2020.

Edited by: Rajni Mohana
Special issue on: Sentiment Analysis and Affective computing in Multimedia Data on Social Network
Received: Apr 25, 2023
Accepted: Oct 16, 2023