Development and Implementation of a Spoken Question Answering System for Bangla Using Large Language Models

by

Shafin Islam Ohin
21241049

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Shafin Islam Ohin
21241049

Approval

The thesis titled "Development and Implementation of a Spoken Question Answering System for Bangla Using Large Language Models" submitted by

1. Shafin Islam Ohin(21241049)

Of Fall 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on February 3, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Farig Yousuf Sadeque
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

This study explores the development and implementation of a spoken question-answering system for Bangla, using the latest advancements in deep learning. To achieve this, this research addresses two key aspects: text-based QA and spoken QA. For the text-based phase, we fine-tuned and evaluated several LLMs including mBERT, Bangla-BERT, RoBERTa on the SQuAD_bn datast. We also evaluated the performance of GPT-4o, Llama 3 by calculating Zero-shot and Few-shot performance. Notably, the GPT-4o with some limitations achieved state-of-the art results on this dataset by outperforming the existing models. A detailed error analysis revealed the limitations was from the dataset inconsistencies. Facing the lack of a Bangla spoken QA dataset, we created a synthesized dataset called Spoken_SQuAD_bn, derived from the SQuAD_bn dataset using the Google Cloud Text-to-Speech API. We benchmarked this new dataset using Automatic-Speech-Recognition (ASR) followed by LLMs, using the Audio Overlapping Score (AOS) metric along with the EM and F1. It showed a significant performance drop because of the ASR error propagation, highlighting the challenges of spoken QA in Bangla. This work establishes a foundation of Bangla spoken-QA by demonstrating the potential as well as the limitations of LLMs in this domain and provides a valuable benchmark dataset for future works.

Keywords: Spoken Question-Answering System; Bangla Language; Deep Learning; ASR; LLM; Low-Resource Languages

Table of Contents

List of Figures

Chapter 1

Introduction

The field of artificial intelligence and natural language processing has come a long way. There are many research works and advancements in question-answering tasks, especially extractive question answering, as it has a lot of real-life use cases. However, Spoken Question Answering (SQA) is still a challenging task even for the most advanced models. The real challenge for Spoken question-answering lies in the task itself. It is "spoken" thus containing audio contents that the current state-of-the-art LLMs couldn't even recognize until very recently SpeechLLMs became a thing. It is still an emerging field and most of the spoken tasks rely on Automatic Speech Recognition (ASR) transcripts that the LLMs can understand. It is even more challenging for low-resource languages like Bengali which continuously struggle to cope with the advancements being made. The main focus of this study is to develop, implement, and show the challenges of a spoken question-answering system for Bangla, a language that 240 million people around the world speak as their first language. We also used a similar approach by using the ASR transcription followed by LLM as shown in Fig. 1.1. Therefore, there were two main steps that we followed. Firstly, as we needed the best-performing model on text-based question answering, we evaluated the squad_bn dataset using the latest large language models to get the best possible score. We also did an in-depth error analysis of this process to find out why the best-performing model failed in the examples that it predicted wrong. Surprisingly, we found that it was mostly due to the dataset's inconsistency rather than

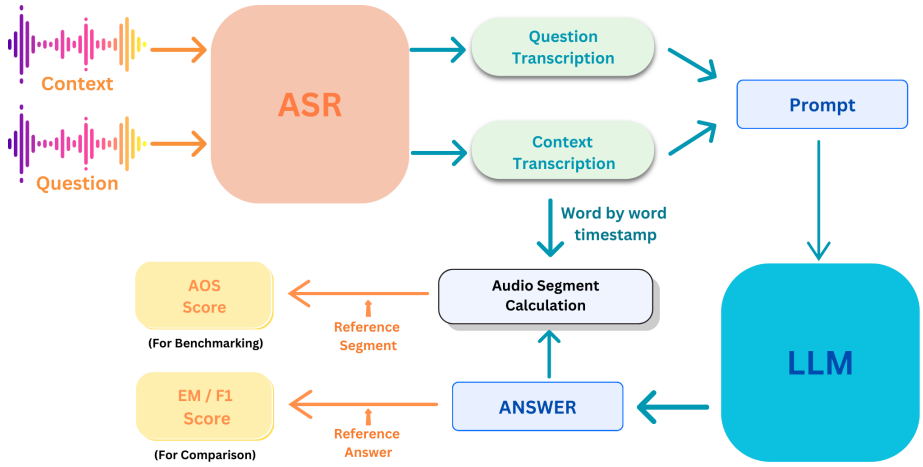


Figure 1.1: Pipeline for the process

the model's limitation and thus it was inevitable. Secondly, we needed a spoken question-answering dataset in Bangla which unfortunately didn't exist and we had to create a synthetic one from the squad_bn. We created it and also benchmarked it using ASR + LLM strategy using proper evaluation metrics. To find the best-performing model on squad_bn, we fine-tuned mBERT, BanglaBERT, RoBERTa on the squad_bn's training data, and then evaluated using the validation data, where BanglaBERT marginally outperformed the others while also getting a steeper loss reduction. It was already reported on the BanglaBERT paper and the score aligned with their report. Then we leveraged the power of recent cutting-edge LLMs and evaluated their validation data using Zero-Shot and Few-Shot learning on LLaMA 3.2 3B, GPT-4o-mini, and GPT-4o. GPT-4o (few-shot) outperformed the other two models and gave far better scores than the previous fine-tuned models. As of our knowledge, this is the current best score on this dataset. Although it performed surprisingly well, it was not perfect. To get a deeper understanding of this gap between the model's score and the perfect score, we did a thorough error analysis by manually checking each of the wrong predictions and tried to build a reasoning behind them. Also, generative LLMs like GPT has a tendency to be creative and use their own knowledge to generate answers. As a result, we had to tune the prompt in a way such that it tries to answer from the given context instead of generating it from its own language knowledge. The few-shot strategy helped in this manner to instruct the model using several examples to follow. We compared the performance of models on squad_bn with our synthetic spoken-squad_bn dataset. The dataset is created using Google Cloud Platforms (GCP) Text-to-Speech (TTS) API. We leveraged the Speech Synthesis Markup Language (SSML) to properly generate the audio contents while keeping track of the answer spans for our new dataset. The evaluation metric we used to benchmark the dataset is a slightly modified version of Audio Overlapping Score (AOS) which is from the paper of Lee C. H. et al. [2] where they created a similar spoken dataset from the original SQuAD dataset. A slight modification is needed because, in our dataset, there are many unanswerable questions. Their proposed metric to calculate AOS doesn't take into account the answerable questions as their dataset doesn't have any. Finally, we got a decent score from the spoken dataset, but not as good as the textual squad_bn dataset. Although we evaluated using the same model that we used for the textual dataset, this degradation of performance shows the impact of ASR error on our spoken dataset. This issue can be only resolved if we can somehow avoid the intermediate step of text generation and use an end-to-end spoken approach which we are leaving as a future work.

1.1 Research Gaps

Spoken QA in Bangla is still a very unexplored field. The main reason is the difficulty due to limited resources. Currently, there is no available dataset on spoken QA in Bangla. The study's goal is to implement a cutting-edge deep learning approach to create a spoken QA system for Bangla and introduce a new challenge in this field by creating a synthetic dataset.

1.2 Research Objectives

- Properly evaluate the existing squad_bn dataset:

  Evaluate the squad_bn dataset by fine-tuning models like mBERT, BanglaBERT, etc, and performing Zero-shot, few-shot on LLaMA 3, GPT-4o-mini, GPT-4o, etc.

- Perform error analysis:

  Do a proper in-depth error analysis on the best-performing model to find out why it failed on the examples it failed. Find out if it is a limitation of the model or the dataset.

- Develop and Optimize a Multi-Stage Pipeline for Spoken Bangla QA:

  A cascade of ASR and LLM can work as a pipeline for Spoken-QA task. OpenAI's whisper or Google's Speech-to-text can be used to generate the ASR transcriptions. Then the generated text can be used with the mBERT, Bangla-BERT, RoBERTa, LLaMA 3, and GPT-4o.

- Create and benchmark a Spoken-QA dataset for Bangla

  Using Google Cloud's text-to-speech API, convert the SQuAD_bn to a spoken dataset. Perform proper testing to make sure the dataset is consistent and error-free. Use or create proper evaluation metrics and benchmark the dataset.

- Fine-Tune and Evaluate the Question Answering Model:

  Use the transcribed text to Fine-tune the encoder models like mBERT, Bangla-BERT, RoBERTa or evaluate with Zero-Shot or Few-shot on Llama 3, GPT-4o-mini, GPT-4o.

- Show that it is possible and has an effect on languages with few resources:

  By performing each step properly, show that it is possible to gain good scores in low-resource languages like Bangla.

1.3 Report Organization

This report is structured into several chapters to provide a clear and systematic presentation of the research.

- Chapter 1 introduces the research problem, objectives, and scope of the study.
- Chapter 2 reviews relevant literature and existing works in the field.
- Chapter 3 describes the dataset preparation and processing techniques.
- Chapter 4 details the models used, their architecture, and the training process.
- Chapter 5 presents the evaluation strategy and experimental results.
- Chapter 6 discusses error analysis and key findings.
- Chapter 7 concludes the report with a summary of contributions and future directions.

Chapter 2

Related Work

2.1 Survey Methodology

The core of this thesis is based on a detailed survey of recent literature on the advancements of extractive question answering and the spoken question answering in Natural Langauge Processing (NLP). Our approach in this thesis also requires a thorough understanding of Automatic Speech Recognition (ASR), thus studying the current advancements in that field. The proper selection of the related articles is very crucial for the correct understanding of the current state of research, finding the gaps and contributing to the ongoing works.

- Identifying Relevant Sources:
  The first step is to identify and pick renowned and credible sources of scientific literature in NLP and Spoken Question Answering. We choose several key repositories and conferences including the ACL (Association for Computer Linguistics) Anthology, NuerIPS (Conference on Neural Information Processing Systems), and ICLR (International Conference on Learning Representations). All of them are well-known and renowned for publishing high-quality research papers and plays an important role in finding pivotal discoveries in the field of Machine Learning.

- Selection Criteria:
  The first selection criteria was very simple - relevancy. If the findings of a paper are not relevant, it's not selected. Then comes the year of publication and citation count. For this thesis, only the relevant papers from the last two to three years were considered.

- Selection Process:
  The initial search resulted in a collection of 21 papers from the selected sources. Each paper was initially reviewed to ascertain its relevance to the topic of extractive question answering. The review process consisted of examining the abstracts, methodology, results, and conclusions to ensure relevance with the research objectives of this thesis. Papers were then filtered out based on the established criteria of publication year and citation count. Then 8 papers were selected, all of which were published in 2022 or 2023 (with one or two exceptions) and had a significant citation count. This chosen set of papers provided a focused and relevant resource for further analysis.

2.2 Related Papers

Through a groundbreaking work led by Pranav Rajpurkar et al. [1], even before the era of transformers, the SQuAD (Standford Question Answering Dataset) dataset became available and had a great impact in the field of machine learning, especially in natural language processing tasks. It's a dataset of extractive question-answering in English. It contains over 100,000 questions created from 536 Wikipedia articles. Each question has a context from where the answer is found. As a large dataset, it not only provides a decent scale for training models but also incorporates a real-world use case where a piece of information needs to be extracted from a large corpus. The creators ensure there is a large diversity among the questions making the reasoning and understanding of the models more challenging. The benchmarks used logistic regression and neural network models which showed a significant performance gap between machine and human by getting a 51% F1 score using the best models compared to 86.8% F1 from human performance. This performance gap creates an opportunity for continuous improvements in the field of Natural Language Processing making it a pivot for future research and innovation in language comprehension.

Two years After the initial success and impact of the Stanford Question Answering Dataset (SQuAD), Pranav Rajpurkar et al[3] created SQuAD 2.0 by addressing the limitations of the SQuAD dataset. It's an upgraded and even more challenging version of the original SQuAD dataset. In the older SQuAD 1.1 dataset, it contained all answerable questions - meaning the answer to each question could be exactly found inside the given context. However in SQuAD 2.0, they introduced around 50,000 new questions, which look valid and answerable, but the answer is not found in the context associated with them, making them unanswerable for this extractive task. This made the task more realistic. Now models not only need to find the correct answer for the questions from the context but also be able to predict correctly if they are not answerable from the given contexts. It enforces the model to learn more about the situation and not just use basic matching methods between the context and question. For example, the model needs to figure out confidently if the context doesn't contain the answer anywhere in it, improving its language understanding. A lot of people manually contributed to making the SQuAD 2.0 dataset. They needed to carefully think out these answerable questions to make sure they were valid and made sense with the context but had wrong or no answers. The target of this dataset is to fool the models that mostly relied on keyword matching and simple text patterns. Several works already report that the SQuAD 2.0 is much harder and more challenging than the original SQuAD. For example, a strong neural network got 86% F1 score in SQuAD but managed to get only 66% F1 in the new dataset while the human performance on this new dataset is 89.5%. This large drop in performance showed the incapability of the methods used earlier with certain answers for each question. As a result the release of this SQuAD 2.0 dataset led to more new ideas in the development of Question Answering systems. It made better models that can perform better in real-world scenarios. It also pushed the limit of what AI could do to understand the interpret human language.

Continuing the exploration of unanswerable questions in the context of SQuAD 2.0,

Vagrant Gautam et al. [6] presented a novel approach to generate such questions automatically. In their paper, the authors proposed a very unique way of adding unanswerable questions to existing QA datasets by using antonyms of words or entity swaps. This approach takes the help of already existing context and answerable questions to generate the unanswerable questions from them. In this process, the requirement for computing resources decreases significantly thus reducing the cost. This process is also much less complex than other well-known approaches. It also improves the performance of the models on datasets like SQuAD 2.0 and TydiQA. This work shows the importance of unanswerable questions while training a model because in real-life cases, there are often many questions that are unanswerable from a given context. They used a relatively simple but clever way where they swapped words with antonyms or switched entities of an answerable question to make it unanswerable. For example, changing "native" to "foreign" in a question that asks about native mammals creates another version of that same question that is unanswerable. They also showed that this simple augmentation approach not only performs better than other complicated ones but also generates easier-to-read questions and also more related to the context. The close relation but unanswerability makes the task much more challenging for the models and enforces a deeper understanding of the context. In addition to that, the models trained on this augmented data performed better with higher F1 scores. This simple approach from the authors shows how important it is to add augmented data for training a model to match with real-world scenarios.

Xun Yao et al. [12] introduced another novel framework called Perturbation for Alignment (PFA) which improves the token embeddings in extractive question-answering (EQA) tasks. They managed to fill an existing gap in the ongoing research which is, not looking at the effects of downstream tasks when the token embedding gets changed. There are two main parts of the PFA framework: semantic alignment and embedding perturbation. The embedding perturbation module uses a multi-layer perceptron (MLP) to change the token embeddings. It changes the initial embeddings to make sure that the model can work in different situations. Then the semantic alignment module aligns the representation of original and changed embeddings to make them semantically consistent. Distance measures like Wasserstein distance are used in this alignment step. This helps in keeping the robustness of the embeddings. The authors also evaluated the models using PFA on datasets like SQuAD, HotpotQA, NewsQA, and NaturalQ. The PFA significantly enhanced the performance of the state-of-the-art models with higher scores. For example, the F1 score increased from 90.3 to 92.4 in the SQuAD dataset. PFA framework works better in low-resource environments and also while applied to other areas. This shows that it is flexible and strong. Their study shows that acknowledging and properly handling the differences in token level makes the model more stable and accurate for further downstream tasks. As a result, PFA works as a big step forward in making reliable extractive question-answering systems.

Mayeesha T. T. et al. [4] showed an innovative approach to developing a question-answering (QA) system for the Bengali language. They raised concerns about the lack of any large, high-quality Bengali Question-Answering dataset and highlighted the importance and necessity of it. As a solution to this problem, they created a syn-

thetic dataset from SQuAD 2.0 by using Google Cloud Translation API and translating a large part of the SQuAD 2.0 dataset from English to Bengali. They used this synthetic dataset called squad_bn for training and testing their benchmarking models. They also evaluated a small human-annotated Bengali question-answering dataset which they created from well-known Bengali Wikipedia articles so that they are related to the culture and context of the area. This paper highlights the use of cutting-edge transformer models on both one-shot and fine-tuned approaches. The models are - multilingual BERT (m-BERT), RoBERTa, and DistilBERT. The zero-shot models were pre-trained on multiple languages including Bengali, and then directly tested on the translated squad_bn dataset. This suggests that cross-lingual transfer learning is also possible. And fine-tuning the models on the training data of squad_bn made them perform even better. However the authors mentioned their critical limitation on machine power and computing resources. Another important factor here is, that they used fuzzy matching techniques to fix the translation mistakes, especially to keep the quality of the answer in the context. This significantly reduced the effect of any bad translation and corrupted data. The comparison with human performance mentioned in the paper makes it a standard for future works in Bengali question-answering systems. The authors were the first to use this method for Bengali and opened a scope to build and work on a low-resource language like Bengali.

The key motivation behind this thesis work is another synthetic dataset from the original SQuAD which can be found in the paper of Lee et al [2]. called Spoken-SQuAD. They explored the challenges of machine listening comprehension, especially focusing on the Automatic Speech Recognition (ASR) errors in transcription. The Spoken-SQuAD is an extraction-based spoken question-answering task. It involves answering a question from an audio context. The answer to that question is a timespan in the audio file of the context where the answer is spoken. This is a realistic evaluation of listening comprehension in comparison to the traditional text-based question-answering tasks. Along with the new synthetic dataset, they also used a new evaluation approach called Audio Overlapping Score (AOS) that considers the impact of ASR errors on spoken question-answering (SQA) tasks. This metric penalizes the model for predicting a wider timespan while rewarding it for overlapping with the reference timespan. As the performance of existing state-of-the-art models of text-based QA tasks declined significantly on the ASR transcripts, the authors proposed a solution to mitigate this error by the inclusion of subword units, data augmentation and confidence scores from the ASR system. The most effective was subword units which break the word into smaller phonetic components for better understanding. Because it caused much fewer phoneme-level errors and allowed the model to grasp the overall meaning more properly. Through these, the paper highlighted the crucial link between accurate speech recognition and effective language understanding. Although it emphasized a better ASR system as it was the main bottleneck for the pipeline, the authors suggested an end-to-end speech model that might be available in the future might outperform all the ASR-based approaches.

The domain of extractive question-answering was dominated by discriminative models like BERT until Mallick P. et al. [8] published their work showcasing the performance of generative models like BART or T5 on the EQA task. As previously

discussed, extractive question answering (EQA) involves finding the exact answer segment to a question from a given context (like in SQuAD). The discriminative models are effective but struggle in some cases where the answer is comprised of a very small fraction of the text or separated into several segments in the text. The authors propose a novel approach that utilizes the generative models by making them predict the indices of the context tokens or sentences that form the answer. This method mitigates the challenges of discriminative models for the case when the answer is spread into several places by allowing the context-aware generation of indices mapped to the parts of the answers. They performed evaluations on multiple QA datasets including MultiSpanQA, BioASQ, MASHQA, and WikiQA, and showed that the generative approach outperforms existing state-of-the-art discriminative models. This paper shows that, as the generative method can model the likelihood of answer positions, it can effectively handle the sparsity found in EQA tasks and thus works as a better alternative to traditional discriminative models. Due to their creative nature, generative models had a bad reputation as information extractors. This work opened new scopes for applying generative models in the areas where discriminative models were thought to be the best performers. These findings encourage the application of generative models in structured output predictions where discriminative models face challenges due to label sparsity and multi-span selections.

Another great contribution in the Spoken question-answering domain is made by You et al. [5] in their paper where they introduced a novel task called Spoken Conversational Question Answering (SCQA). The main goal of this task is to build models capable of handling multi-tern conversations based on spoken documents. As the founder of this task, the authors created a dataset called Spoken-CoQA. The dataset contains around 40,000 question-answer pairs from approximately 4,000 audio conversations. The dataset is designed for evaluating models on SCQA systems in dialogue-style interactions by focusing on the challenges due to the difficult integration of speech and text information. To mitigate this challenge, the authors proposed a unified data-distillation approach called DDNET which incorporates cross-modal information (from Audio and text) to gain a fine-grained representation of speech and language. They also introduced a mechanism called dual attention which makes the alignment of audio and text more accurate and makes the knowledge transfer less error-prone. Their experimental result showed that the existing state-of-the-art models performed significantly badly on the Spone-CoQA dataset suggesting the necessity of effectively incorporating cross-modal information. Their proposed DDNET method got a much superior performance in the spoken conversational QA task.

The challenges of cascading ASR and LLMs are another domain to explore. Labrak Y. et al. [13] focused on these challenges inherent in traditional spoken question-answering (SQA) systems. The conventional approach where ASR transcription is fed to the LLM, suffers from cumulative errors and higher computational cost because of their multi-stage process. To overcome this, the authors proposed an end-to-end solution that completely removes the necessity of intermediate transcription and directly processes the spoken questions to generate answers in the medical domain. This approach uses LLMs to handle the tasks. They evaluated the Zero-shot SQA framework and compared the performance with traditional cascading systems.

The evaluation was on an open benchmark with eight medical tasks and 48 hours of synthetic audio data. Their proposed approach reduces the resource requirement by 14.7 times compared to a cascade of 1.3 billion parameter LLM and 1.55 billion parameter ASR model while gaining 5% improvement in average accuracy. These findings highlight the potential of end-to-end methodologies in the spoken domain while showcasing the drawbacks and limitations of traditional cascading systems.

Jian Wu et al. [11] introduced a novel approach to performing speech processing using Large Language Models (LLMs) called Specch-LLaMA. In traditional methods, a cascading strategy is used for this task where Automatic Speech Recognition (ASR) models create a text transcription of audio,o and then LLMs process that text input. As previously discussed, this multi-stage pipeline has a lot of issues inherent to it. However, Speech-LLaMA avoids this intermediate text generation and directly maps the acoustic features into the LLM's continuous semantic space. This approach makes the processing much smoother and way faster. At the heart of this method, there lies an audio encoder and a Connectionist Temporal Classification (CTC) compressor. The compressor is needed because the audio embedding is many times longer than a text embedding and is impossible to directly feed into any LLM. The CTC compressor compresses or shortens the speech signals and makes them compatible with shorter text signals. After the compression, the audio signals are turned into vectors in the LLM's semantic space. Because of this direct integration with the audio, now the LLMs will be able to handle the transcription or translation of sound information much better. In the experiments done by the authors, speech-to-text tasks in multiple datasets including CoVoST with 13 languages showed large improvements over the baseline models. The proposed Speech-LLaMA model not only got a higher BLEU score but also was parameter efficient. As a result, it is much better than other encoder-decoder architectures.

Chapter 3

Dataset

3.1 SQuAD_bn

This project uses the Bangla version of the SQuAD 2.0 dataset, which was translated by the BUET CSE NLP Group. The SQuAD 2.0 dataset is used as a standard for extractive question answering. It has questions that can be answered based on the given context or can't be answered. The original dataset's structure and complexity have been kept in this translation to Bangla. This makes it a useful tool for building and testing question-answering models in Bangla.

3.1.1 Dataset Description

The dataset is organized into three splits: train (118k), test(2.5k), and validation(2.5k). In the training dataset, as shown in Fig 3.1 the length of contexts is spread out so that most are between 1 and 300 words, with a peak between 60 and 70 words. There are no words in the context at all, and there are 600 words at most. This distribution is right-skewed and has only one mode, which means that shorter contexts are much more common than longer ones. To get the most out of model training and performance, we need to understand this distribution. Since there are fewer words for the model to process, shorter contexts often lead to more accurate answer extraction. The fact that shorter contexts are more common suggests that users usually ask questions about short passages in real life. Also, the fact that there are contexts with no words shows how important it is to deal with edge cases like questions that can't be answered. This distribution helps with planning the preprocessing steps, choosing the right model architectures, and setting the training parameters so that the model works well with different lengths of context.
Questions in the dataset range from 2 to 35 words, with most questions being 6 to 8 words long. This distribution in Fig 3.1 is also right-skewed and unimodal, suggesting that shorter questions are more common.
The lengths of answers vary between 0 to 25 words. The distribution shown in 3.1 forms a J-shape, continuously decreasing, which means that shorter answers are much more frequent than longer ones. This distribution is right-skewed as well.
The starting positions of answers within the context range from 0 to 3000 characters according to the distribution in Fig 3.1. This distribution is also J-shaped and right-skewed, indicating that many answers start at the beginning of the context. It is notable how many answers actually have a start position of 0, suggesting that

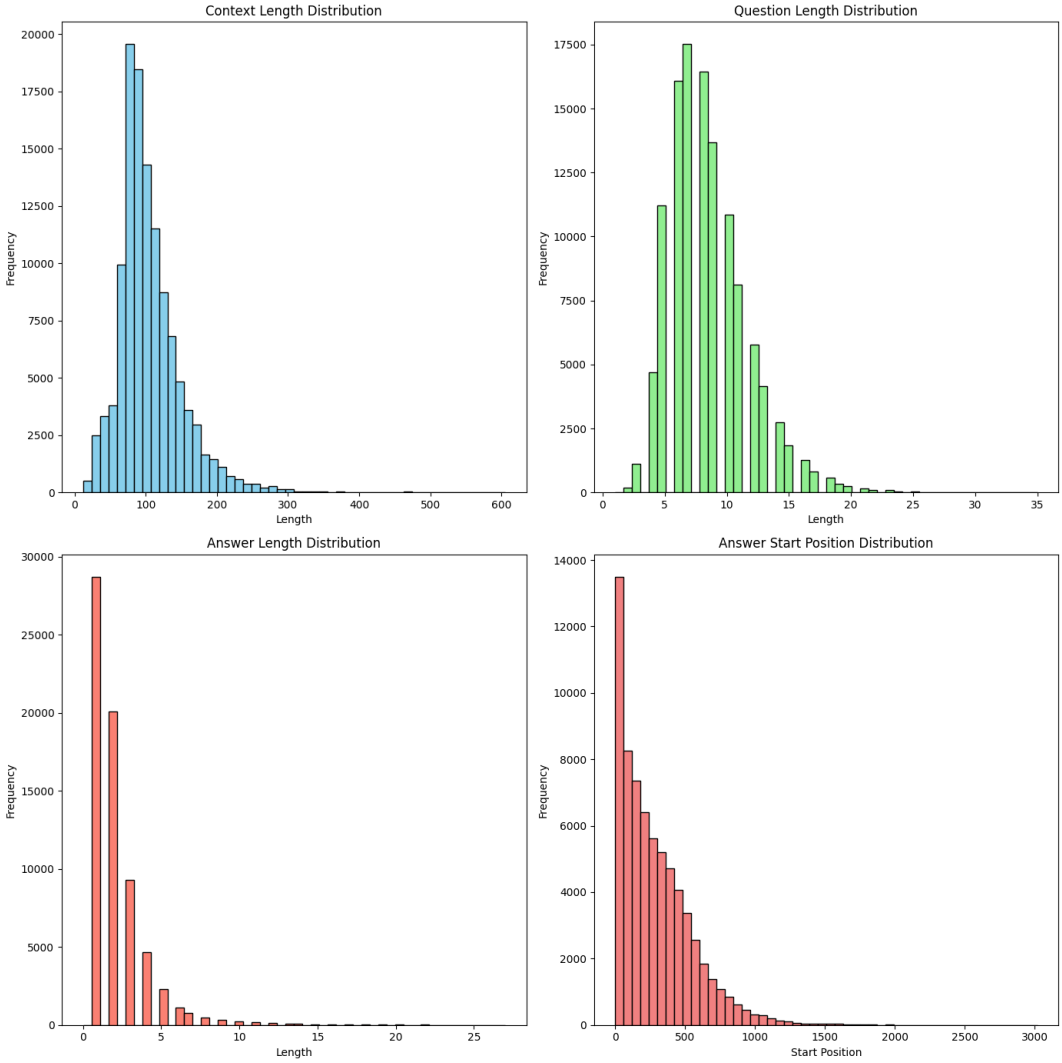


Figure 3.1: Histograms of text lengths

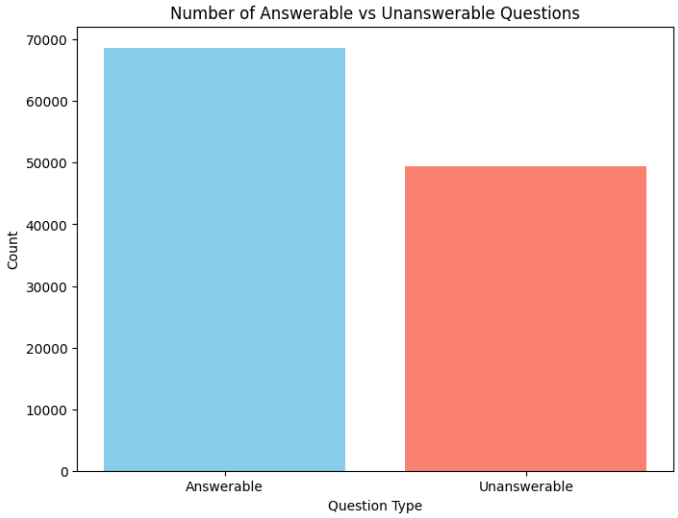


Figure 3.2: Number of Answerable questions vs Non answerable questions

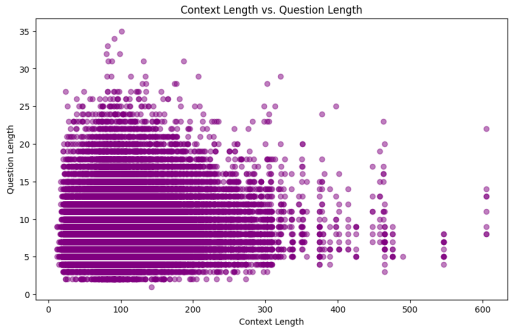
Figure 3.3: WordCloud for most frequent words (Excluding stopwords)

relevant information often appears at the start of the context.

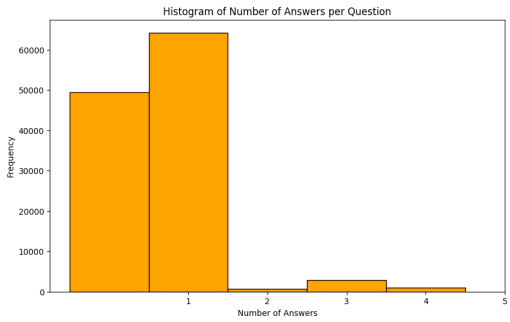
A significant aspect of the SQuAD 2.0 dataset is the inclusion of unanswerable questions, which test the model's ability to identify when a question cannot be answered based on the given context. In this dataset, there are 68,674 answerable questions and 49,443 unanswerable questions as shown in Fig 3.2. This balance ensures that the model learns to avoid guessing answers for unanswerable questions, an essential capability for robust question-answering systems.

The WordCloud of the context shows that the most common words are সালে, সালের, সাল, নামে, পরিচিত, বছর, মার্কিন, যুক্তরাষ্ট্রের, অন্যান্য, ভিত্তি, গঠিত as shown in Fig 3.3. This suggests that, there is a significant amount of presence of years, names and most of the contents are related to USA. If we look at the source of the dataset, it justifies the reason.

An analysis of the relationship between context length and question length shown in Fig 3.4a indicates that as the context length increases, the question length slightly decreases. However, the correlation is very weak, indicating that question length is largely independent of context length.



(a) Context length vs Question Length



(b) Number of answers per Question

Figure 3.4: Comparison of different dataset characteristics

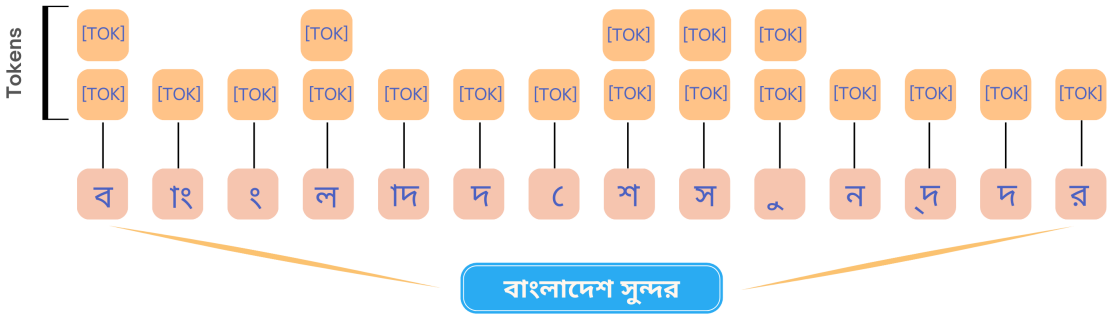


Figure 3.5: Tokenization of a sentence

The distribution shown in Fig 3.4b of the number of answers per question in the dataset reveals that most questions have only one answer. However, there is a significant number of questions that have zero answers, indicating they are unanswerable from the given context. A much smaller proportion of questions have two, three, or four answers. This distribution highlights that the majority of questions have a clear, concise answer, while a substantial portion of questions are designed to be unanswerable. Only a small number of questions require multiple answers for completeness, which reflects the structure of the dataset aimed at training models to handle both answerable and unanswerable queries effectively.

3.1.2 Data Preprocessing

In this section, we describe the preprocessing steps applied to the dataset to prepare it for model training. Preprocessing is a critical phase in any machine learning pipeline, ensuring that the raw data is transformed into a format suitable for the model to learn from.

Tokenization and Encoding The preprocessing function preprocess_function leverages a tokenizer to transform the questions and contexts into tokenized inputs that the model can understand. Here's a step-by-step breakdown of the preprocessing process:

1. Extracting Questions and Contexts: Each example in the dataset contains a question, a context, and a set of answers. The preprocessing function first extracts these components.

2. Tokenizing Inputs: The tokenizer processes the questions and contexts, converting them into token IDs while also applying truncation and padding to ensure uniform input length.

    - **Truncation**: Ensures that the tokenized inputs do not exceed the maximum length of 1024 tokens.
    - **Padding**: Pads the inputs to the maximum length, ensuring all inputs are of the same length.

3. Generating Start and End Positions: For each question-context pair, the function calculates the start and end positions of the answer within the tokenized context. We use offset mapping returned from the tokenizer to perform this action. The original input data has the answer_start positions which is the character level starting index of the answer in the context. But, while tokenizing, a chunk of characters can be represented by a single token, therefore the answer_start position gets invalid for the tokenized data. So we need offset mapping to recalculate the answer_start position and also the answer_end position. It sounds very straightforward, but for a low-resource and complex language like Bangla, the tokenization is still not up to the mark. As a result, a single word gets divided into several tokens, even a single character is sometimes represented by two or more token as we can see in 3.5. As a result, recalculating the answer_start and answer_end position is troublesome. There is no single function that can do this correctly for every answer. So, we used a very customized function that does a satisfactory job recalculating the answer start and end positions in the tokenized ids.

The preprocessing function is applied to all splits of the dataset (train, validation, and test). This processes each example in the dataset with the defined preprocessing function and removes unnecessary columns. By applying this preprocessing pipeline, we ensure that the dataset is in a format suitable for training a question-answering model. Each example is tokenized, padded, and has the start and end positions of the answers accurately mapped, providing the model with all necessary information to learn and make predictions effectively.
This structured work plan provides a clear roadmap for the research, ensuring a focused approach towards achieving the research objectives within the stipulated 10-month timeframe.

3.2 Spoken-SQuAD_bn

3.2.1 Dataset Creation

We used Google Cloud Platform's (GCP) text-to-speech API to generate the spoken dataset. As previously mentioned, the training dataset has 118117 question-context pairs. But, there are 20080 unique contexts with repetitions. The questions are unique. As the GCP TTS API is a paid API, we didn't pass each question-context pair separately, rather, generated the context Audios first while keeping track of which question belongs to which context. Then we generated the question audios. The same strategy was applied for validation and test data also, but as they don't

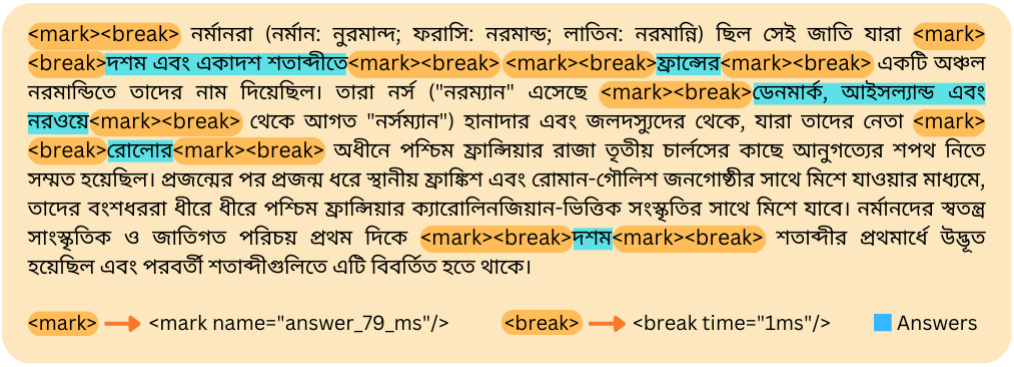


Figure 3.6: Answer Markings of a context

contain repetitive context, it wasn't required for them. The task can be divided into two main tasks - preprocessing the text data and actual synthesis.

Preprocessing

Before sending the text to the TTS API, we need to properly format it to make the audio quality better and produce clear audio. The preprocessing task can be broken down into several steps -

- **Answer Positions Marking:** In extractive question answering, the answer to each question is a text span in the context. The same should apply to spoken datasets where the answer should be an audio span in the context. But, if we convert the context directly to an audio file, we will lose that positional information for the answer that is present in the context. So, we took the help of Speech Synthesis Markup Language (SSML) where we can include named <mark> tags inside the context and the timestamps for the tags will be returned by the API that can be used to get positional information for the answer. But, as previously mentioned, there are multiple questions present for the same context. This means, that while generating the audio for a single context, we need to keep track of multiple answers that are present in the context. So, we need to enclose each answer with a start mark tag and end mark tag to get both the start_time and end_time of the answer. But it created a complicated issue for us, which can easily go unnoticed. For several contexts, there are multiple answers that fully or partially overlap each other. In those cases, the mark tags can be consecutive and much harder to insert at the correct position. Even if they are placed properly without any issue, the GCP's API ignores all of them if it finds any consecutive mark tags. To handle this issue, we took a set of all the answer's start_positions and end_positions for a context. So they became unique indices that we needed to keep track of. Then we inserted marks in all those positions and named each of them according to their indices so that we could map them to the desired answers. Finally, to be more on the safer side, we included time breaks using <break> tags after each mark tag so that even if they get side by side, there is always a time break between them. This solved the issue of consecutive mark tags as shown in fig 3.6. But there was another serious issue. The Bengali language has an abugida writing style (alphasyllabary) where the vowel and consonant often join together to make a single unit. This can be a critical issue here

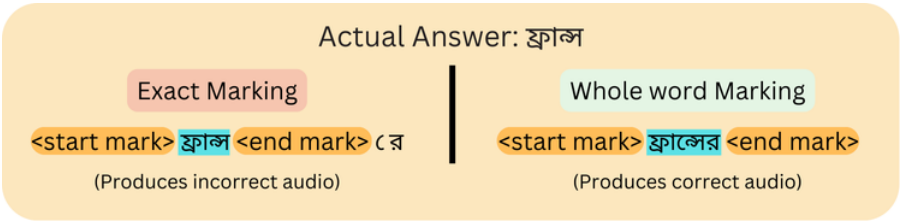


Figure 3.7: Exact Marking vs Whole Word Marking

while marking the answer positions as it might break the word into segments. To solve this issue, as shown in fig 3.7, we took the entire word as part of the correct answer whenever the answer ended abruptly inside the word.

- **Data cleaning and SSML generation:** Each text passed through a function that cleaned the text and converted it into Speech Synthesis Markup Language (SSML). Although the context already contains mark and break tags which are the syntax of SSML, it is not considered as SSML until entirely enclosed by <speak> tags. We had to mark the answers before the text cleaning process because otherwise, we would have lost the answer positions as they are represented by indices. Therefore any change in the context makes the answers lost in it. For the cleaning process, we used regex to perform the search and replace option by pattern matching. We had to perform several cleaning steps to achieve a decent and working SSML text.
  - **SSML preserved characters:** First we replaced the characters that are preserved by SSML using proper syntax for them. We changed "&" into "&amp", and "'" into "'" by directly replacing them. We also had to change the less_than (<) with "<" and greater_than (>) with ">". But there were already <mark> and <break> tags present in the context. So, we couldn't just search and replace them blindly. We used regex to find out all the less_than (<) with "<" and greater_than (>) symbols except the <mark> and <break> tags, and replaced them with "<" and ">".
  - **Random symbols and spaces:** We removed all the 3 or more consecutive symbols (\, ?, '(', ')', etc) as they don't contain any information for audio and might confuse the speech model. We also removed all the consecutive spaces (2 or more spaces together) to avoid unnecessary pauses.
  - **Adding SSML tags:** We have added some SSML tags for a better audio output. We enclosed each year written in "yyy" or "yyyy" format with <say-as> tags to pronounce as a date. This makes the model read them like a date instead of just reading out the digits sequentially.
  - **Replacing "-" to "থেকে":** In several places of the contexts, the dash "-" symbol is present between two numbers (especially between years, ex: "১৯৭১ - ১৯৭৫") and the speech model ignores it completely. To solve this issue we replaced all such occurrences of "-" with "থেকে" so that the speech model reads it properly.
  - **Footnote markers removal:** The contexts of the validation dataset of squad_bn are from Wikipedia Bangla articles. They contain a lot of

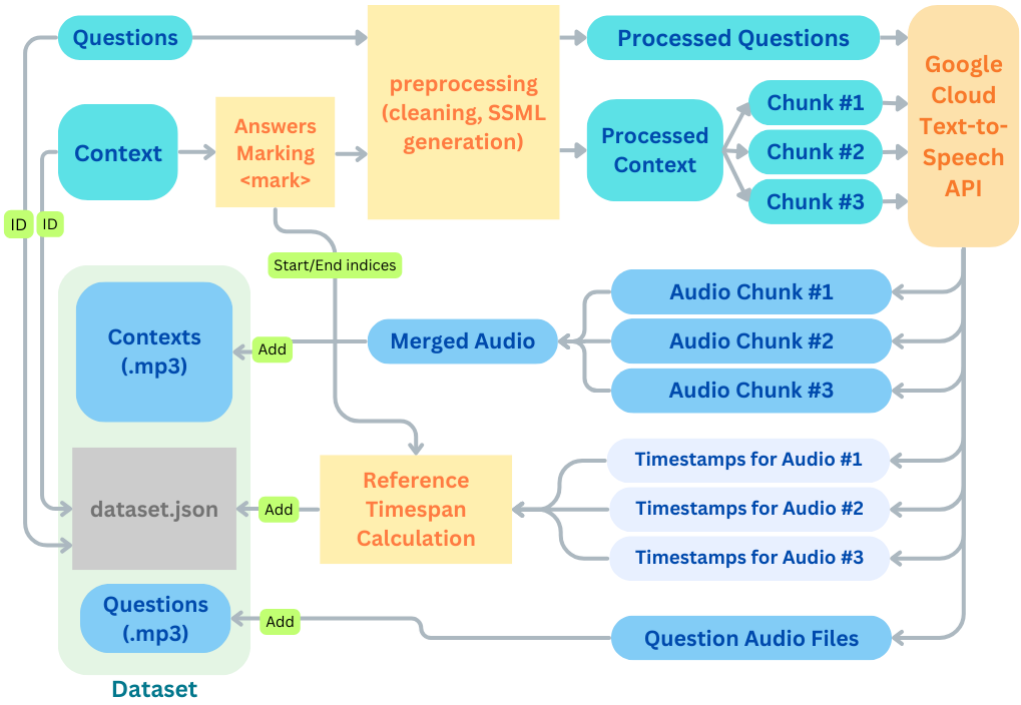


Figure 3.8: Pipeline for dataset generation

footnote markers in square brackets (ex: "[2]") and were not removed while creating the squad_bn dataset. They were not much of an issue in the text dataset, but while converting to audio, the speech model reads out the numbers which is not intended at all. So, we replaced all such footnote markers from the text with empty strings using regex.

Synthesis

After preprocessing a text, it is ready to send to the API for audio conversion. But, the Google Cloud Platform's TTS API has a 5000-byte limit per request. And, many contexts in the dataset cross this limit because the Bangla text is in utf-8 format and it needs 3 bytes per character to store. So we had to split and synthesize the text if it crossed the limit of 5000 Bytes. However many SSML tags exist in the text, which makes it hard to split without breaking the tags. To get around this issue, we split the entire text on Bangla full-stop ('।') and got the list of sentences. This made sure that no SSML tags were broken while splitting. Then we concatenated the sentences one by one until they were about to cross the 5000-byte limit. This gave us a usable chunk to send to the API and the corresponding audio chunk. We repeat this process until all the chunks of the entire text are finished processing. Then we merged the audio chunks to generate the actual output audio in "mp3" format. While splitting and merging, we needed to keep track of the timestamps properly to avoid misalignment. Then we used these timestamps of the SSML marks to generate the reference timespan for each of the answers present in that context. The questions were processed similarly and converted to audio. The dataset generation pipeline is shown in Fig 3.8.

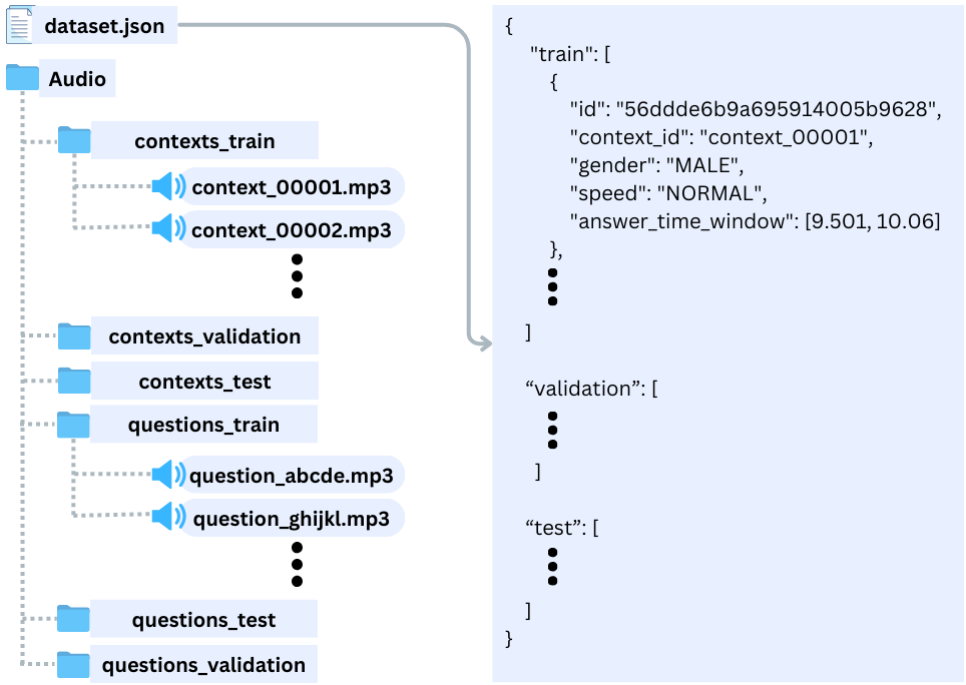


Figure 3.9: Directory structure of the dataset

3.2.2 Dataset Description

The dataset consists of audio files in ”mp3” format of the contexts and the questions. The structure of the dataset as shown in Fig 3.9 contains 1 folder called ”Audio” and a metadata file called ”dataset.json”. The audio folder has 6 folders in it, named ”contexts_train”, ”contexts_validation”, ”contexts_test”, ”questions_train”, ”questions_validation”, and ”questions_test”. Each of these folders contains the audio files that the name suggests. The context audio files are named ”context_00001.mp3”, ”context_00002.mp3” etc. The filename (except the extension) represents the ”id” of that context which is used in the metadata file. Similarly, the questions folders also contain the audio of the questions named like this - ”question_bengali–6120429524876336-0-2651.mp3”. Here the string between ”question_” and ”.mp3” (bengali–6120429524876336-0-2651) is that question’s id which is used in metadata. The metadata file contains the question and context mapping information along with some other information for each of the dataset split.

As we can see in the Fig 3.9, for each question, the metadata info contains the corresponding context’s id as ”context_id”. The gender field denotes the gender of the voice of that question and context. There are two values for this field, ”MALE” and ”FEMALE”. The dataset is created with around half male voices and half female voices to introduce diversity. Also, the ”speed” field denotes the speaking rate of the audio with 3 different values - ”SLOW”, ”NORMAL”, and ”FAST” each having equal portions as shown in Fig 3.10. This also adds some variety to the dataset and helps generalized learning for models. The proportions are maintained for different speeds among both genders. The male voices have equal portions of each of the three different speeds just like the female voices as seen in Fig 3.11.

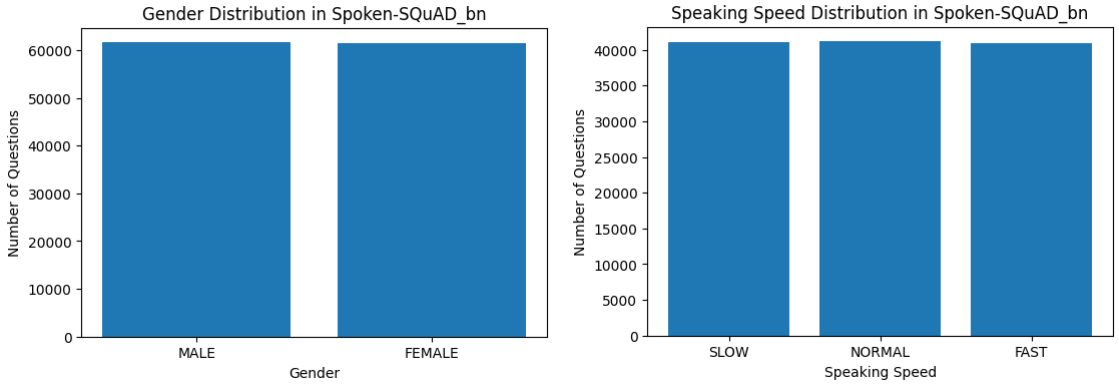


Figure 3.10: Distribution of both genders and different speaking rates

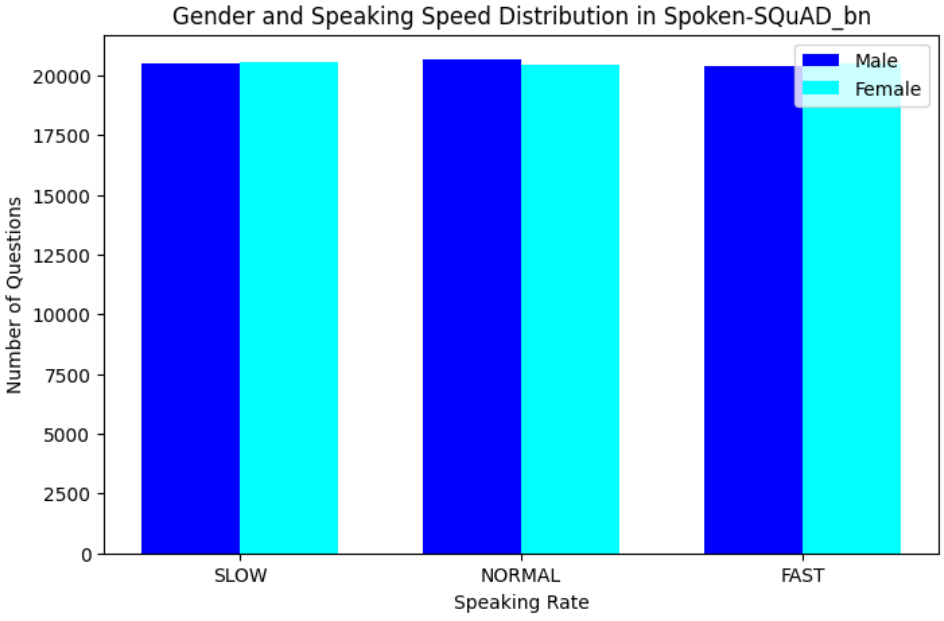


Figure 3.11: Count of Male and Female on different speaking rates

Chapter 4

Models

4.1 Overview

This chapter talks about the models we used in our project, focusing on their structure, how they were trained, and how they were prompted. The models we used for the thesis are mBERT, RoBERTa, BanglaBERT, Llama 3.2 3B, GPT-4o-mini, and GPT-4o. The the text-to-speech and speech-to-text we used the cloud platform's API of google.

4.2 mBERT (Multilingual BERT)

4.2.1 Model Architecture

mBERT is a multilingual understanding model that follows the Transformer architecture, with a single model architecture and shared vocabulary across 104 languages. It consists of either 12 or 24 encoder layers, each with self-attention mechanisms (multiheaded attention: 12 or 16) and feed-forward networks for input sequence processing as shown in Fig 4.1. mBERT uses WordPiece tokenization, with a vocabulary size of about 110,000 tokens, along with positional and token-type embeddings. It is pre-trained on Wikipedia data using masked language modeling and next-sentence prediction objectives to learn contextualized word representations, then fine-tuned for a variety of downstream tasks.

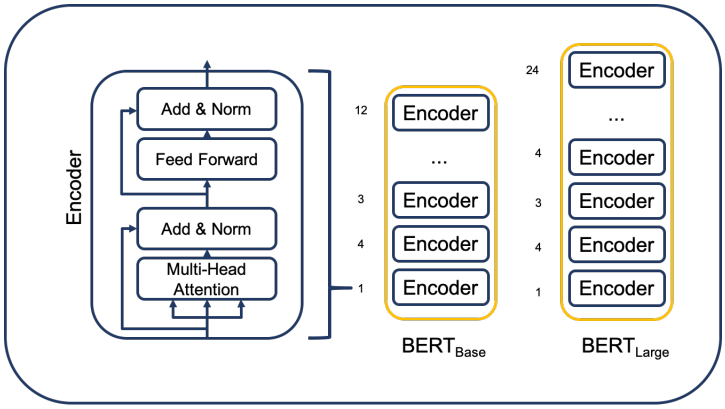


Figure 4.1: BERT Architecture [14]

4.2.2 Training Parameters and Setup

We used a learning rate of 3e-5, and weight_decay of 0.01 while training the mBERT model. We trained it for 2 epochs using 16bit floating point representation and the default optimizer AdamW.

4.3 xlm-RoBERTa-large

4.3.1 Model Architecture

xlm-RoBERTa-large is also a transformer-based model but utilizes a larger architecture with 24 layers, 16 attention heads, and a hidden size of 1024 as shown in Fig 4.2. It builds upon the RoBERTa architecture and is trained on a massive multilingual dataset using a cross-lingual masked language modeling objective. This approach allows xlm-RoBERTa-large to learn more robust and nuanced representations compared to mBERT, leading to improved performance on a wide range of cross-lingual NLP tasks.

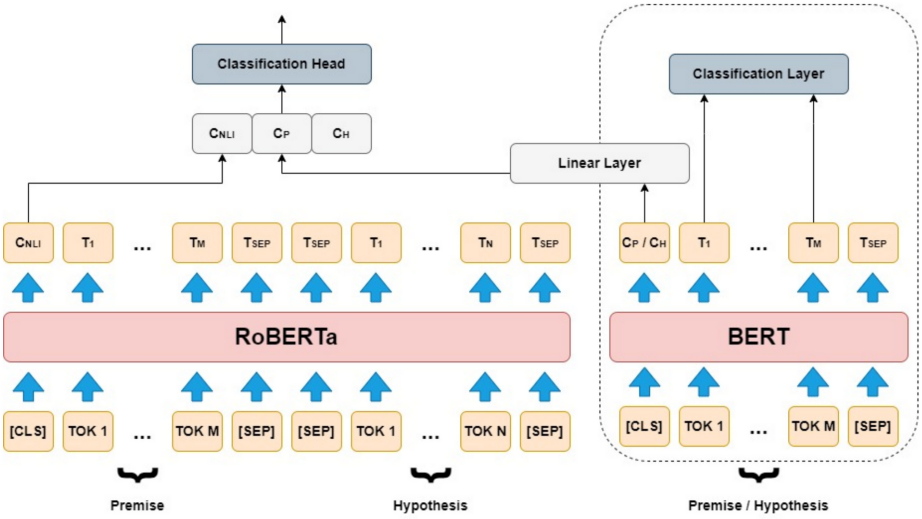


Figure 4.2: RoBERTa Architecture [7]

4.3.2 Training Parameters and Setup

We used a learning rate of 3e-5, and weight_decay of 0.01 while training the mBERT model. We trained it for 2 epochs using 16bit floating point representation and the default optimizer AdamW.

4.4 Bangla-BERT

4.4.1 Model Architecture

BanglaBERT is a transformer-based model specifically pre-trained for the Bengali language. It typically follows the BERT architecture as shown in Fig 4.3 with variations in the number of layers, attention heads, and hidden size depending on the

specific implementation. BanglaBERT is trained on a large Bengali corpus, enabling it to capture the unique characteristics and nuances of the language. This specialization allows BanglaBERT to achieve state-of-the-art results on various Bengali NLP tasks, outperforming multilingual models like mBERT and xlm-RoBERTa-large in many cases.

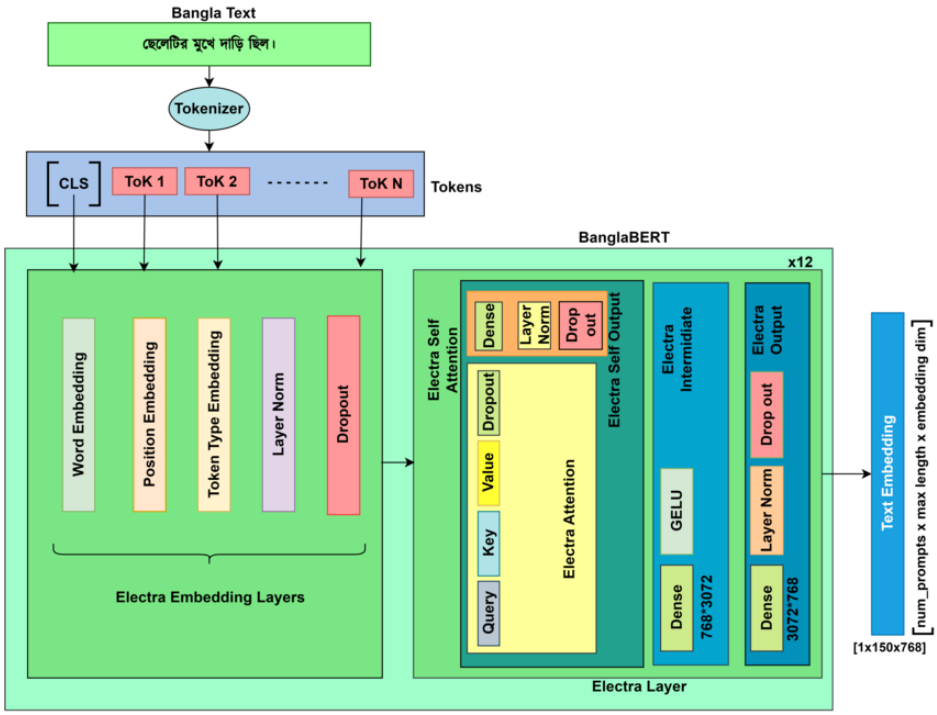


Figure 4.3: BanglaBERT Architecture [10]

4.4.2 Training Parameters and Setup

We used a learning rate of 3e-5, and weight_decay of 0.01 while training the mBERT model. We trained it for 2 epochs using 16bit floating point representation and the default optimizer AdamW.

4.5 Llama 3.2 3B

4.5.1 Model Architecture

Llama 3.2 3B is an auto-regressive language model that utilizes an optimized transformer architecture. It features a 3.21 billion parameter model size, making it lightweight and suitable for on-device applications. The model employs Grouped-Query Attention (GQA) for improved inference scalability and supports a context length of up to 128,000 tokens. Llama 3.2 3B was pretrained on a massive dataset of 9 trillion tokens of public multilingual text and code, incorporating knowledge distillation from larger Llama 3.1 models (8B and 70B). It has been fine-tuned for instruction following and chat applications using techniques like Supervised Fine-Tuning (SFT) and Reinforcement Learning with Human Feedback (RLHF) to align with human preferences for helpfulness and safety. 1 This model excels in tasks such

as text summarization, translation, and knowledge retrieval, especially in multilingual settings. The architecture of this model is shown in Fig 4.4.

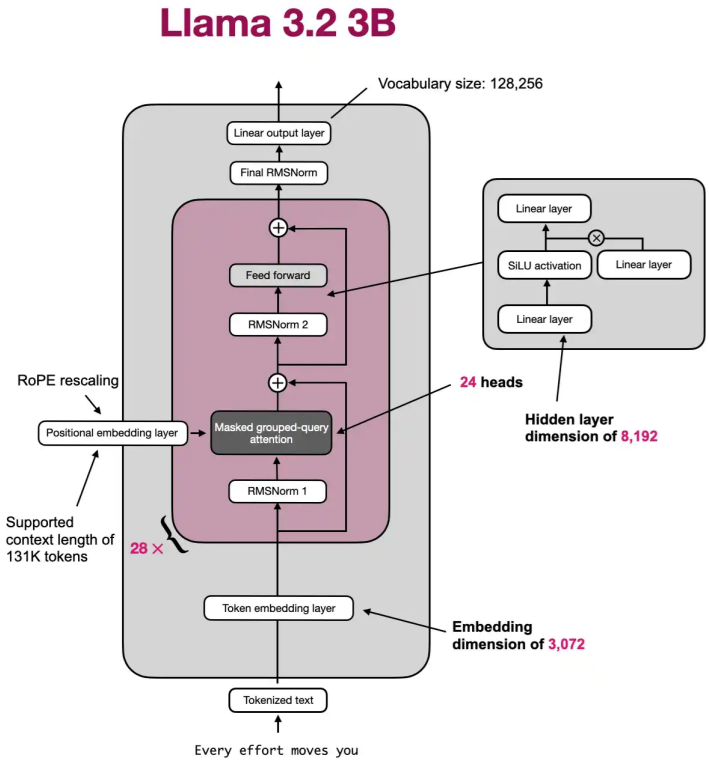


Figure 4.4: Llama 3.2 3B architecture [9]

4.6 GPT 4o

4.6.1 Model Architecture

GPT-4's architecture is based on a transformer model, but with a significant increase in scale and complexity compared to its predecessors. It reportedly utilizes a Mixture-of-Experts (MoE) architecture, where multiple smaller expert models specialize in different tasks or data types. This allows GPT-4 to handle a wider range of inputs and generate more nuanced and accurate outputs. While the exact number of parameters and layers in GPT-4 remains undisclosed, estimations suggest it has around 1.8 trillion parameters, distributed across 120 layers and 16 expert models. Each expert model has approximately 111 billion parameters, and only two are activated for each inference query, making the process more efficient. This massive scale, combined with a larger training dataset and advancements in training techniques, contributes to GPT-4's improved performance on various tasks, including language understanding, code generation, and image analysis.

4.7 Fine-Tuning approach

4.7.1 Data Preprocessing

- **Dataset:** SQuAD-BN contains (context, question, answer) triples in Bengali, where some questions might not be answerable from the given context.
- **Tokenization:** Use the appropriate tokenizer for each model (mBERT's tokenizer, XLM-RoBERTa's tokenizer, BanglaBERT's tokenizer). Tokenize the context and question, typically concatenating them with a special separator token (e.g., [CLS] question [SEP] context [SEP]).
- **Input Formatting:** Input IDs: Convert tokens to their corresponding IDs in the model's vocabulary.
  Attention Mask: Create an attention mask to distinguish real tokens from padding tokens.
  Token Type IDs: If using them (typically done for BERT models), differentiate between the question segment and context segment using token type IDs (e.g., 0 for the question, 1 for the context).
- **Target Formatting:** Start/End Positions: For answerable questions, identify the start and end positions of the answer within the tokenized context.
  Unanswerable Questions: For unanswerable questions, you might set both the start and end positions to a special token's index (e.g., the [CLS] token index, which is often 0). This indicates that the answer is not present in the context.

4.7.2 Model Adaptation

- Load Pre-trained Model: Load the pre-trained weights of the chosen model (mBERT, XLM-RoBERTa-large, or BanglaBERT).
- Add a Question-Answering Head: On top of the last hidden state of the pre-trained model, add a linear layer (or a small multi-layer network) that will be trained to predict the start and end positions of the answer. This layer typically has an output dimension of 2 (one for start, one for end).

4.8 Training

- Loss Function: Used a combination of cross-entropy losses: Start Position Loss: Cross-entropy loss between the predicted start logits and the true start position. End Position Loss: Cross-entropy loss between the predicted end logits and the true end position.
- Optimizer: AdamW was used
- Learning Rate Schedule: The learning rate scheduler (linear decay) was used to gradually decrease the learning rate during training.

- Batch Size: Used the maximum possible batch size that the hardware could handle
- Epochs: Trained for 2 epochs over the SQuAD-BN dataset.
- Early Stopping: Monitored the performance on a validation set and stopped training if the performance started to degrade to prevent overfitting.

4.9 Zero-shot and Few-shot prompts

4.9.1 Zero Shot

The "system" prompt was: "Use the given Context to find the answer to the question in Bangla. Just write the answer part, not a complete sentence. If the answer is not present in the context, write "NO ANSWER". The "user" prompt was: "Context: (context was here) \n\n Question: (question was here)"

4.9.2 Few Shot

For the Few shot evaluation, the system prompt remained the same as the Zero shot. But the user prompt contained 4 examples of context-question-answer. The examples are -
context_1 = "বাংলাদেশ ১৯৭১ সালে সাধীন হয়।"
ques_1 = "বাংলাদেশ কত সালে সাধীন হয় ?"
ans_1 = "১৯৭১"
context_2 = "বাংলাদেশে ২০২৪ সালে একটি বিপ্লব সংঘটিত হয়। এটি জুলাই বিপ্লব নামে পরিচিত।"
ques_2 = "জুলাই বিপ্লব কবে সংঘটিত হয় ?"
ans_2 = "২০২৪ সালে"
context_3 = "রমজান মাসকে তিন ভাগে ভাগ করা হয়।"
ques_3 = "রমজান মাসকে কয় ভাগে ভাগ করা হয় ?"
ans_3 = "তিন"
context_4 = "বাংলাদেশে জেলা ৬৪ টি।"
ques_4 = "বাংলাদেশে জেলা কয়টি ?"
ans_4 = "৬৪"

The "user" prompt: "Here are few example contexts, questions and answers. (examples). Now use the context to find the answer to the question.\n Context: (context) \n\n Question: (question)"
For both the Zero-shot and few-shot, the model returns a chat completion content that contains the model's predicted answer for that particular context.

Chapter 5

Evaluation and Results

Here we show the performances of the models on squad_bn and spoken-squad_bn dataset. We evaluated the squad_bn using the Exact Match (EM) and F1 score. They are the standard metrics for the SQuAD 2.0 dataset. For Spoken-squad_bn, we used the Audio Overlapping Score (AOS) metric.
The Training-loss vs Step graph for the fine-tuning of mBERT, xlm-RoBERTa-large, and BanglaBERT is shown in Fig 5.1. They almost flattened down at the end of the second epoch, suggesting no need for further training.

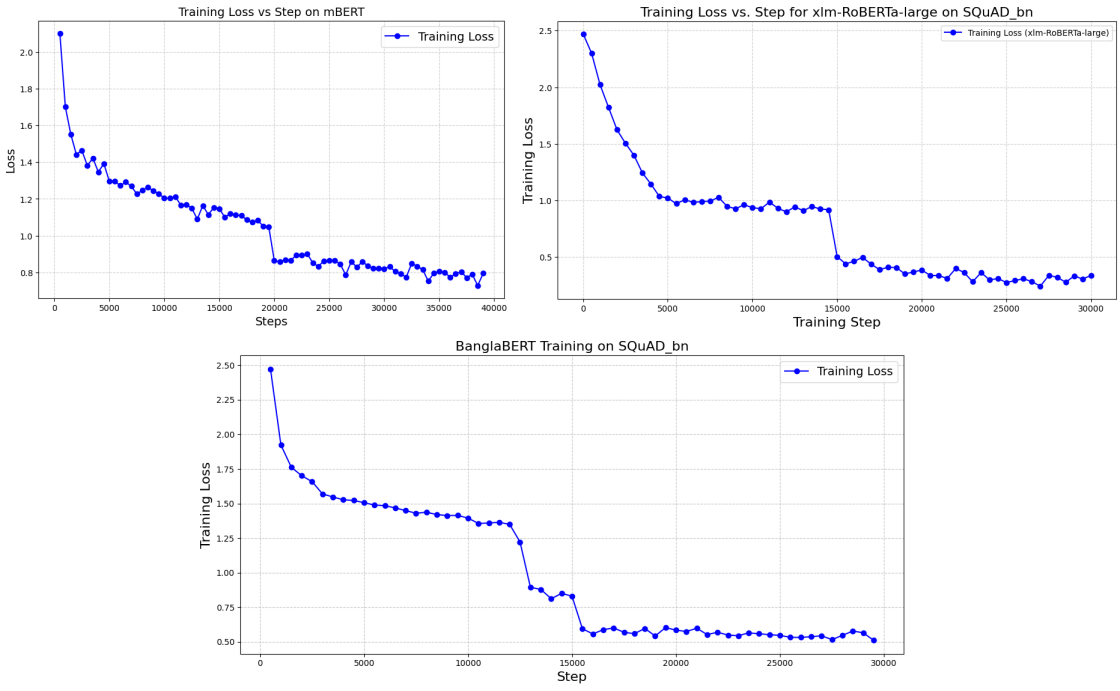


Figure 5.1: Training Loss vs Steps graph

5.1 Evaluation Strategy

The metrics we used are -

- **Exact Match (EM):** It gives 1 if the predicted text is completely the same as the reference text. Otherwise, it gives 0. To calculate this, we just compared

	Original Answer	Predicted Answer
Text	ফ্রান্স	ফ্রান্সের
Tokens	ফ, ্র, া, ন, ্স	ফ, ্র, া, ন, ্স, ে, র
Precision	$\frac{4}{4+1} = 0.8$	
Recall	$\frac{4}{4+0} = 1$	
F1	$\frac{2*0.8*1}{0.8+1} = 0.8889$	

Table 5.1: F1 Calculation of an example

the two strings. This score represents how capable a model is to predict the exact answer span from the context.

- **F1 Score:** Just missing the answer span by 1 index can result in a 0 in the EM score. But, we also need to know how close the model can predict to reference answers. Thats where the F1 score helps. It doesn't completely reject an answer even if it is slightly different from the reference answer. Two calculate the F1 score of a predicted answer for a reference answer, we tokenized both of them. Then, the precision and recall is calculated from the predicted tokens and reference tokens. Which gives us the F1 score. A single example of calculating EM and F1 is shown in Table 5.1.

- **Audio Overlapping Score (AOS):** This metric is only applicable for the spoken dataset. It gives the overlapping score of two timespans, one is the reference timespan and the other is the predicted timespan. This metric was originally proposed by Lee et al. [2], however, we used a slightly modified version of it so that it can correctly calculate the scores for the unanswerable cases. The formula is -

$$AOS = \frac{(X \cap Y) + 1}{(X \cup Y) + 1}$$

where the X is the predicted timespan and the Y is the reference timespan in milliseconds. The +1's are for the case when there is no answer and both sets are empty and it should return 1. For other cases, a perfect overlap will return 1, and partial overlap will return a value less than 1 but greater than 0, and in case of no overlap, it will return a value very close to 0.

5.2 Results

5.2.1 Performance on SQuAD_bn

The F1 and EM scores for the models on squad_bn are shown in Table 5.2. GPT-4o outperformed the other models in the Few-Shot evaluation with an Exact Match of 75.18 and F1 of 85.89. The performance of this model is self-explanatory given the size and scale of this model. BanglaBERT also produced a really good score in both Exact Match and F1. If we look at the F1 score, we can see that the generative LLMs are good at giving an answer close to the original answer thus getting a much

Model	EM	F1	HasAns_EM	HasAns_F1	NoAns
mBERT	67.95	72.83	40.45	50.20	95.44
xlm-RoBERTa-large	70.04	75.24	51.40	62.05	87.96
BanglaBERT	71.22	76.57	**53.00**	63.68	89.45
LLama 3.2 3B (Zero Shot)	53.16	60.35	50.28	64.67	56.04
Llama 3.2 3B (Few Shot)	55.27	63.12	52.44	67.30	58.85
GPT-4o-mini (Zero Shot)	68.75	78.00	41.73	60.24	95.77
GPT-4o-mini (Few Shot)	73.06	82.25	47.32	65.70	**98.80**
GPT-4o (Zero Shot)	72.02	83.45	50.20	73.06	93.85
GPT-4o (Few Shot)	**75.18**	**85.89**	52.04	**73.45**	98.32

Table 5.2: Performance of different models on squad_bn

Dataset	EM	F1
SQuAD_bn	75.18	85.89
Spoken-SQuAD_bn	59.88	70.15

Table 5.3: Performance comparison of GPT-4o (Few-Shot)

higher F1 scores compared to the encoder models. But, when it comes to predicting the answer exactly as it is, they seem to struggle. Therefore we can see a lower EM score for GPT-4o-mini compared to BanglaBERT or RoBERTa. Although GPT-4o got a slightly better F1 due to its scale. In the error analysis chapter, the main reason behind the lower EM score of GPT-4o is analyzed thoroughly with proper examples.

5.2.2 Performance on Spoken-SQuAD_bn

We only evaluated the performance of GPT-4o on Spoken-SQuAD_bn as it was the best-performing model on the squad_bn. The AOS metric is valid only for the standalone speech processing models that can directly take speech as input and understand it to produce output without the help of any pre-trained LLM. But, due to the unavailability of such a model, we used a cascading approach of ASR and LLM to achieve speech understanding. Therefore, we had to calculate the AOS score using the LLM's text output by mapping it with the audio to generate something like a "pseudo-output" of the audio span. We use the term "pseudo-output" instead of just "output" because it is not the actual output that the model produces. Rather, we are calculating this audio span prediction using the model's text output. Therefore, it is an output calculated from real output. It serves the purpose well, but still, it is a fake output. Because we are actually getting text output from the models, we also calculated the EM and F1 metrics to compare with the text dataset squad_bn. The EM an F1 score of GPT-4o in few-shot evaluation on Spoken-SQuAD_bn is shown in Table 5.3. We can see a great downfall in both scores for the same model. This is due to the ASR error and is unavoidable in the cascading system.
Moreover, the Spoken-SQuAD_bn dataset has a little diversity as previously discussed in the dataset chapter. It contains male, and female voices with slow, normal, and fast speaking rates. The EM, F1, and AOS score for each type of speech is listed in Table 5.4. The scores show that a faster speaking rate gives a lower performance

Speed \ Gender	ALL	MALE	FEMALE
ALL	59.88/70.15/53.19	60.24/70.31/53.37	59.52/70.00/53.00
SLOW	61.90/72.17/54.79	61.40/71.79/53.91	62.41/72.56/55.67
NORMAL	60.43/70.86/52.21	62.83/72.19/53.14	58.03/69.52/51.29
FAST	57.31/67.42/52.57	56.49/66.94/53.07	58.13/67.90/52.06

Table 5.4: Performance (EM/F1/AOS) of GPT-4o (few-shot) on Spoken-SQuAD_bn

as it is hard to comprehend for the ASR model. However, there is no such clear difference in scores between male and female voices, suggesting that gender bias is not present in the ASR model.

Chapter 6

Error Analysis

In this section, we analyze the errors of GPT-4o (Few Shot) on the squad_bn dataset to find out the reason behind a much lower Exact Match (EM) score than F1. To do this, we manually checked the examples from incorrect predictions and tried to build some reasoning on why the model might have produced the incorrect predictions. In this process, we found several inconsistencies in the dataset. They are described with proper examples below -

- **Issue while answering a date:** When asked "কত সালে ?" There are two types of answers in the dataset. One includes the word "সালে" in the answer and another type is just the year.
  Example:
  In the 56th validation data,
  Context: "...নামক এক ভদ্রলাকের নেতৃত্বে ১৮৯৯ সালে একদল সুইস..."
  Question: "...কত সালে...?"
  Reference Answer: "১৮৯৯"

  But in the 97th validation data,
  Context: "...২০০০ সালের মধ্যেই এই সিরিজটি..."
  Question: "...কত সালে...?"
  Reference Answer: "২০০০ সালে"

  A similar case can be seen when asked "কবে ?". There are also two types of answers found for it.
  Example:
  In the 27th validation data,
  Context: "...করেছিলেন। ১৮৩৩ সালে মেনিনজাইটিসে আক্রান্ত হয়ে..."
  Question: "...কবে...?"
  Reference Answer: "১৮৩৩ সালে"

  But in the 307th validation data,
  Context: "...বিজেপির উৎস ১৯৫১ সালে পশ্চিমবঙ্গের জন্মদাতা..."
  Question: "...কবে...?"
  Reference Answer: "১৯৫১"

  We can see that there is no way to learn to predict all of them correctly. If the

model learns to exclude the word "সালে" when asked "কত সালে ?", it will produce incorrect output for examples like the 97th validation data. Therefore, in this case, learning to be correct in some examples, makes it predict incorrect answers for others. This is one of the main issues behind the incorrect predictions and lower EM score because a large amount of question involves asking about dates.

- **More accurate answers:** There were several cases where the model predicted a correct but more accurate answer that is present in the context. However, the dataset only listed a less accurate answer as the reference answer. Here are some examples -
  In the 69th validation data,
  Context: "...সুই সম্রাট ইয়াং (৫৬৯ – এপ্রিল ১১, ৬১৮) ছিলেন সুই সম্রাট ওয়েনের দ্বিতীয় পুত্র এবং সুই সাম্রাজ্যের দ্বিতীয় সম্রাট। তার পারিবারিক নাম ছিল ইয়াং গুয়াং, অন্য নাম ইয়াং, ডাকনাম আমাে, এবং তার নাতী ইয়াং তংয়ের সংক্ষিপ্ত রাজত্বকালে সম্রাট মিং নামে পরিচিত ছিলেন।..."
  Question: "সুই সাম্রাজ্যের দ্বিতীয় সম্রাটের নাম কী ?"
  Reference Answer: "ইয়াং"
  Model Predicted: "ইয়াং গুয়াং"
  The model's prediction is correct and present in the context. It should've been the reference answer.

  In the 265th validation data,
  Context: "...১৯৭১ সালের ২৫ মার্চ রাতের অন্ধকারে পাকিস্তানি সামরিক বাহিনী পূর্ব পাকিস্তানে বাঙালি নিধনে ঝাঁপিয়ে পড়লে একটি জনযুদ্ধের আদলে গেৱলিাযুদ্ধ তথা সাধীনতা যুদ্ধের সূচনা ঘটে।..."
  Question: "বাংলাদেশের মুক্তিযুদ্ধ কবে শুরু হয় ?"
  Reference Answer: "১৯৭১ সালে"
  Model Predicted: "১৯৭১ সালের ২৫ মার্চ"
  Again the model's predicted answer is more accurate and also present in the context. We can see the dataset has less accurate answers as a reference when more accurate answers are present in the context.

  In the 20th validation data,
  Context: "...যদিও ১৯৯৬ সালের ১২ ডিসেম্বর নতুন দিল্লিতে ভারতের প্রধানমন্ত্রী এইচ. ডি. দেবেগৌড়া ও বাংলাদেশের প্রধানমন্ত্রী শেখ হাসিনা ওয়াজেদ একটি সামগ্রিক বৈদেশিক চুক্তি সাক্ষর করেন..."
  Question: "গঙ্গার জল বণ্টনকে কেন্দ্র ক'রে বাংলাদেশ ও ভারত রাষ্ট্রদ্বয়ের মধ্যে প্রথম কানে সালে চুক্তি সাক্ষরিত হয় ?"
  Reference Answer: "১৯৯৬ সালের ১২ ডিসেম্বর"
  Model Predicted: "১৯৯৬"
  This time, the question specifically asked about the year, and the model predicted it correctly. But, the reference answer contains the entire date including month and day which is not the required information by the question.

  Here are two more examples that shows the accuracy inconsistency -
  In the 135th validation data,
  Context: "...বাংলাদেশের বর্তমান প্রধানমন্ত্রী হলেন শেখ হাসিনা, বাংলাদেশ আওয়ামী

লীগের সভানেত্রী..."
Question: "বর্তমানে বাংলাদেশে কানে রাজনৈতিক দল ক্ষমতায় রয়েছে ?"
Reference Answer: "আওয়ামী লীগ"
Model Predicted: "বাংলাদেশ আওয়ামী লীগ"

And in the 100th validation data,
Context: "মাহোম্মদ আব্দুল জলিল (২১ জানুয়ারি ১৯৩৯ - ৬ মার্চ ২০১৩) ছিলেন একজন বাংলাদেশী রাজনীতিবিদ এবং বিশিষ্ট ব্যবসায়ী। তিনি বাংলাদেশ সরকারের প্রাক্তন বাণিজ্যমন্ত্রী, বাংলাদেশ আওয়ামী লীগের প্রাক্তন সাধারণ সম্পাদক। ..."
Question: "মাহোম্মদ আব্দুল জলিল কানে রাজনৈতিক দলের সদস্য ছিলেন ?"
Reference Answer: "বাংলাদেশ আওয়ামী লীগের"
Model Predicted: "আওয়ামী লীগ"
From these two examples we can see that when asked about the same thing "কানে রাজনৈতিক দল", the reference answers are different. One of them includes the word "বাংলাদেশ" and the other doesn't.

- **Incorrect Reference Answers:** There are several examples for which the given reference answer is not related to the question and consists of a random chunk from the context. The model got 0 in both EM and F1 in these examples. Here are some of them -
In the 13th validation data,
Context: "...তিব্বতের শান্তিপূর্ণ মুক্তির সতেরাে দফা চুক্তি বলতে কার্যতঃ সাধীন তিব্বতের রাষ্ট্রপ্রধান চতুর্দশ দলাই লামার প্রতিনিধিদের সঙ্গে সদ্যপ্রতিষ্ঠিত গণচীন সরকারের ১৯৫১ খ্রিষ্টাব্দে সাক্ষরিত চুক্তিকে বাঝোনাে হয়..."
Question: "তিব্বতের শান্তিপূর্ণ মুক্তির পদক্ষেপের চুক্তি কত সালে সাক্ষরিত হয় ?"
Reference Answer: "সাধীন তিব্বতের রাষ্ট্রপ্রধান চতুর্দশ দলাই লামার প্রতিনিধিদের সঙ্গে সদ্যপ্রতিষ্ঠিত গণচীন সরকারের"
Model Predicted: "১৯৫১ খ্রিষ্টাব্দে"

In the 346th validation data,
Context: "আনুমানিক ৫৭০ খ্রিস্টাব্দে (হস্তিবর্ষ) মক্কা নগরীতে জন্ম নেওয়া মুহাম্মাদ মাতৃগর্বে থাকাকালীন পিতা হারা হন শিশু বয়সে মাতাকে হারিয়ে এতিম হন এবং প্রথমে তাঁর পিতামহ আবদুল মাত্তোলিব ও পরে পিতৃব্য আবু তালিবের নিকট লালিত পালিত হন।..."
Question: "আবু আল-কাশিম মুহাম্মাদ ইবনে □আবদুল্লাহ ইবনে □আবদুল মুত্তালিব ইবনে হাশিম কবে জন্মগ্রহণ করেন ?"
Reference Answer: "মক্কা নগরীতে"
Model Predicted: "৫৭০ খিষ্টাব্দে"

In the 36 validation data,
Context: "...নন্দী নামে একটি পৌরাণিক ষাঁড় শিবের বাহন।..."
Question: "হিন্দু পুরান মতে শিবের বাহন কে ?"
Reference Answer: "ষাঁড়"
Model Predicted: "নন্দী"
When asked "কে ?", the correct answer should be " " which is the name of the bull. "ষাঁড়" would be correct if the question asked "কি ?".

In the 139th validation data,
Context: "কলকাতা বিমানবন্দর ঐতিহ্যগতভাবে ইউরাপে থেকে ইন্দাচৈীন ও অস্ট্রেলিয়ার মধ্যে বিমান রুটে কৌশলগত কেন্দ্র হিসেবে কাজ করে। ১৯৩৭ সালে এ্যামিলিয়া ইয়ারহাট সহ বহু বিমান সংস্থার অনেক অগ্রগামী উড়ানগুলি যাত্রা শুরু করে কলকাতা থেকে। ১৯২৪ সালে, কেএলএল বিমান সংস্থা কলকাতায় তাদের আমস্টারডামের -বতভিয়া (জাকার্তা) রুটের অংশ হিসাবে অবতরণ শুরু করেন। একই বছরে, কানেও বিমান বাহিনী কর্তৃক প্রথম বিশ্ব ভ্রমণ অভিযানের অংশ হিসাবে একটি রয়েল এয়ার ফার্সে বিমান কলকাতায় অবতরণ করে।"
Question: "নেতাজী সুভাষ চন্দ্র বসু আন্তর্জাতিক বিমানবন্দর কবে নির্মিত হয় ?"
Reference Answer: "১৯৩৭ সালে"
Model Predicted: "NO ANSWER"
Here, the context doesn't even contain the name "নেতাজী সুভাষ চন্দ্র বসু আন্তর্জাতিক বিমানবন্দর". Therefore this should be an unanswerable question.

- **Proper noun spelling issue:** In many examples, the answer is a name. But beacuse Bengali fonts has the "Abugida" writing style, the name exists in a way that it has a Bengali vowel attached to its end, specially "ৈ". Here are some examples -
  In the 461st validation data -
  Question: "গণিতে ক্যালকুলাসের জনক কে ?"
  Model's prediction: "আইজাক নিউটন ও গটফ্রিড লাইবনিৎস"
  Reference answer: "আইজাক নিউটন ও গটফ্রিড লাইবনিৎসে"

  Similarly in the 635th validation data, Question: "রামপালের বাবার নাম কী ছিল ?"
  Model's prediction: "তৃতীয় বিগ্রহপাল"
  Reference answer: "তৃতীয় বিগ্রহপালে"

We can see the dataset has many minor inconsistencies that affect the performance of the model especially on Exact Match score. These types of consistencies are almost inevitable while creating a synthetic dataset and require manual checking to be corrected. Nevertheless, it was one of the reasons behind the lower EM score.
On the other hand, there were some limitations from the model's side too. Being a generative model, The GPT-4o generated several answers that were not a direct part of the context. This also reduced the EM score for the generative models. To summarize, there are limitations in both the dataset and the generative models that justify the much lower EM scores compared to F1.

Chapter 7

Conclusion

This thesis has successfully demonstrated the Spoken Question Answering task in Bangla using the current state-of-the-art large language models. It used a synthesized spoken question-answering dataset to achieve this. The thesis work has set a foundational benchmark for future advancements in Bangla spoken-QA systems by introducing the Spoken-SQuAD_bn dataset and benchmarking it with Audio Overlapping Score (AOS) metric. The fine-tuning and evaluation of several models like BanglaBERT and GPT-4o showed how well they can perform on question answering. The significant decrease in performance on the spoken dataset shows the error propagation issue of cascading ASR and LLM models. This limitation can be addressed in future research using end-to-end spoken QA solutions.

Bibliography

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds., Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. [Online]. Available: https://aclanthology.org/D16-1264.

[2] C.-H. Lee, S.-L. Wu, C.-L. Liu, and H.-y. Lee, "Spoken squad: A study of mitigating the impact of speech recognition errors on listening comprehension," in *Interspeech 2018*, 2018, pp. 3459–3463. DOI: 10.21437/Interspeech.2018-1714.

[3] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for SQuAD," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, I. Gurevych and Y. Miyao, Eds., Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 784–789. DOI: 10.18653/v1/P18-2124. [Online]. Available: https://aclanthology.org/P18-2124.

[4] A. M. S. Tasmiah Tahsin Mayeesha and R. M. Rahman, "Deep learning based question answering system in bengali," *Journal of Information and Telecommunication*, vol. 5, no. 2, pp. 145–178, 2021. DOI: 10.1080/24751839.2020.1833136. eprint: https://doi.org/10.1080/24751839.2020.1833136. [Online]. Available: https://doi.org/10.1080/24751839.2020.1833136.

[5] C. You, N. Chen, F. Liu, S. Ge, X. Wu, and Y. Zou, "End-to-end spoken conversational question answering: Task, dataset and model," in *Findings of the Association for Computational Linguistics: NAACL 2022*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds., Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 1219–1232. DOI: 10.18653/v1/2022.findings-naacl.91. [Online]. Available: https://aclanthology.org/2022.findings-naacl.91/.

[6] V. Gautam, M. Zhang, and D. Klakow, "A lightweight method to generate unanswerable questions in English," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 7349–7360. DOI: 10.18653/v1/2023.findings-emnlp.491. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.491.

[7] Labellerr. (2023). "Roberta model architecture." [Online; accessed 2023], [Online]. Available: https://www.labellerr.com/blog/content/images/2023/06/Roberta.png.

[8] P. Mallick, T. Nayak, and I. Bhattacharya, "Adapting pre-trained generative models for extractive question answering," in *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, S. Gehrmann, A. Wang, J. Sedoc, E. Clark, K. Dhole, K. R. Chandu, E. Santus, and H. Sedghamiz, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 128–137. [Online]. Available: https://aclanthology.org/2023.gem-1.11.

[9] S. Raschka. (2023). "Gpt to llama: Llama-32 architecture." [Online; accessed 2023], [Online]. Available: https://github.com/rasbt/LLMs-from-scratch/blob/main/ch05/07_gpt_to_llama/standalone-llama32.ipynb.

[10] ResearchGate. (2023). "Internal layered architecture of banglabert." [Online; accessed 2023], [Online]. Available: https://www.researchgate.net/publication/376181622/figure/fig2/AS:11431281213621055@1703135085437/nternal-layered-architecture-of-BanglaBERT.png.

[11] J. Wu, Y. Gaur, Z. Chen, L. Zhou, Y. Zhu, T. Wang, J. Li, S. Liu, B. Ren, L. Liu, and Y. Wu, "On decoder-only architecture for speech-to-text and large language model integration," Dec. 2023, pp. 1–8. DOI: 10.1109/ASRU57964.2023.10389705.

[12] X. Yao, J. Ma, X. Hu, J. Yang, Y. Guo, and J. Liu, "Towards robust token embeddings for extractive question answering," in *Web Information Systems Engineering – WISE 2023*, F. Zhang, H. Wang, M. Barhamgi, L. Chen, and R. Zhou, Eds., Singapore: Springer Nature Singapore, 2023, pp. 82–96, ISBN: 978-981-99-7254-8.

[13] Y. Labrak, A. Moumen, R. Dufour, and M. Rouvier, "Zero-shot end-to-end spoken question answering in medical domain," in *Interspeech 2024*, 2024, pp. 2020–2024. DOI: 10.21437/Interspeech.2024-1344.

[14] Medium. (2024). "Illustration of a machine learning model pipeline." [Online; accessed 2024], [Online]. Available: https://miro.medium.com/v2/resize:fit:1400/1*9T7r1yXe5zUcobWDQTPSdw.png.