TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A

FINAL REPORT

ON

JHATTA SAMACHAR: LEVERAGING TRANSFORMER MODELS FOR PERSONALIZED NEWS DELIVERY

SUBMITTED BY:

AMRIT SHARMA (PUL077BCT009)

KRIPESH NIHURE (PUL077BCT037)

DARPAN KATTEL (PUL077BCT099)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

SAMSUNG INNOVATION CAMPUS

Nov, 2024

Acknowledgments

We sincerely thank the **Department of Electronics and Computer Engineering** at the Institute of Engineering, Pulchowk Campus, for providing a supportive learning environment and the resources necessary for the successful completion of this project.

Our heartfelt gratitude extends to **Samsung Innovation Campus** for their valuable academic programs. The engaging classes, comprehensive study materials, and practical lab sessions significantly enhanced our knowledge and were pivotal in the development of this project.

We are especially grateful to our Supervisor, **Dr. Arun Kumar Timalsina**, for his expertise, guidance, and constructive feedback, which were instrumental in shaping this project.

We are deeply thankful to our mentor, Sir **Dr.Suresh Pokhrel**, whose expertise and insightful feedback have been invaluable in shaping the direction of this project.

We also appreciate the encouragement and valuable suggestions provided by our peers throughout this journey. Their collaboration and insights played a crucial role in refining our work.

This project represents a collective effort, and we are deeply thankful to everyone who contributed to its success. Your support and involvement have been invaluable.

Abstract

This report details the development and completion of our personalized news summarization application JhattaSamachaar. The project utilizes a finetuned transformer model to generate concise, user-specific news summaries, using a dataset collected and prepared in-house for training purposes. The backend, developed with Django, and the Flutter-based mobile frontend were successfully integrated and deployed. With its seamless functionality and personalized approach, the application delivers a culturally relevant and efficient news experience tailored to user interests.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

The rise of digital media has led to an overwhelming amount of news online, making it hard for users to stay updated on what matters most. News summarization helps by providing short, personalized summaries of the most important information. Our project uses data science and natural language processing (NLP) with transformer models to create meaningful and tailored news summaries. By focusing English news articles, we used established datasets and real-time scraped content for a well-rounded, personalized news experience.

1.2 Problem Statement

In today's fast-paced world, people seek more information in less time. We have observed that even when we listen to a 5-minute news segment on YouTube, only about 1 minute is actual news, while the rest is background information. This not only wastes time but also diminishes the news listening experience.

1.3 Objectives

The main objectives of this project are:

1. To construct or finetune a T5 transformer model for the purpose of summarizing news articles effectively.

2. To gather, preprocess, and utilize extensive datasets for training and validating the summarization model.

3. To create a functional, cross-platform mobile application using Flutter that can present news summaries and allow customization based on user preferences.

4. To integrate the backend, summarization model, and frontend app into a cohesive system, allowing for real-time summarization and deployment on a production scale.

1.4 Scope

The scope of this project spans the full-stack development of a personalized news summarization system. It involves:

1. Data Collection and Processing: Using the scraped news data from Ekantipur for supervised training.

2. Model Development: Finetuning a transformer-based T5 summarization model capable of handling multilingual content in the context of locally prepared dataset.

3. Frontend and Backend Development: Developing a Django backend for API management and data storage, and a Flutter-based mobile application for user interaction.

Out of scope

Once the model is finalized and the system integrated, the final stage will focus on deploying the application and backend in production for user accessibility, which is a future enhancement and currently out of Scope.

2. Literature Review

2.1 Related Work

In recent years, substantial work has been done in the field of text summarization, particularly with the advent of transformer-based models. Traditional extractive methods, such as *TF-IDF* and *Latent Semantic Analysis* (LSA), although effective for smaller datasets, often fail to capture semantic nuances in large and diverse datasets. This has led researchers to focus on neural-based methods, which allow for abstractive summarization, generating summaries that rephrase or condense original text.

2.1.1 Abstractive Summarization Models

Recurrent Neural Networks (RNNs) and Sequence-to-Sequence Models

Earlier approaches for abstractive summarization used RNNs and sequence-to-sequence (Seq2Seq) models, which enabled mapping input sequences to output sequences. For instance, the work by *Nallapati et al.* demonstrated the efficacy of Seq2Seq with attention mechanisms for abstractive summarization. However, RNNs suffer from limitations in capturing long-range dependencies due to their sequential nature, often leading to issues like information loss and lower efficiency.

Attention and Transformer Models

The introduction of the transformer model by *Vaswani et al.* marked a turning point in natural language processing. Unlike RNNs, transformers use self-attention mechanisms that enable the model to weigh the importance of different words in a sentence, improving the model's ability to capture contextual relationships and long-range dependencies. This innovation paved the way for advanced transformer-based models, such as *BERT* and *GPT*, which have been applied across a wide range of NLP tasks.

2.1.2 The T5 Model and Summarization

The **T5 (Text-To-Text Transfer Transformer)** model, introduced by *Raffel et al.*, is a powerful language model designed specifically to treat every NLP problem as a text-to-text task. By framing all tasks in this unified format, T5 can be fine-tuned for diverse applications, including text summarization. The architecture of T5 is based on the encoder-decoder structure of transformers, enabling it to learn bidirectional context on the encoder side and autoregressive decoding.

The T5 model's flexibility has made it especially effective for summarization tasks, as it can adapt to different lengths and styles of text. For instance, T5 has demonstrated state-of-the-art results on benchmark datasets like CNN-DailyMail, a dataset extensively used for summarization research. In the current project, we have chosen T5 as the model architecture for generating summaries due to its advanced ability to understand and generate coherent, contextually accurate summaries.

Challenges with Transformer Models

While T5 and other transformer models have significantly advanced summarization, they come with challenges such as computational requirements and the need for large labeled datasets. Training a model like T5 from scratch requires considerable computational power, which often limits experimentation to environments with GPU support, such as Google Colab and Kaggle. Additionally, transformers have a high memory footprint, especially for lengthy input sequences, which is a notable consideration when handling large datasets.

2.2 Related Theory

2.2.1 Transformer Architecture

Transformers are based on a fully self-attention mechanism, which computes attention scores between all words in an input sequence. This allows the model to understand contextual relationships across the entire sequence in parallel, overcoming the sequential processing limitations of RNNs.

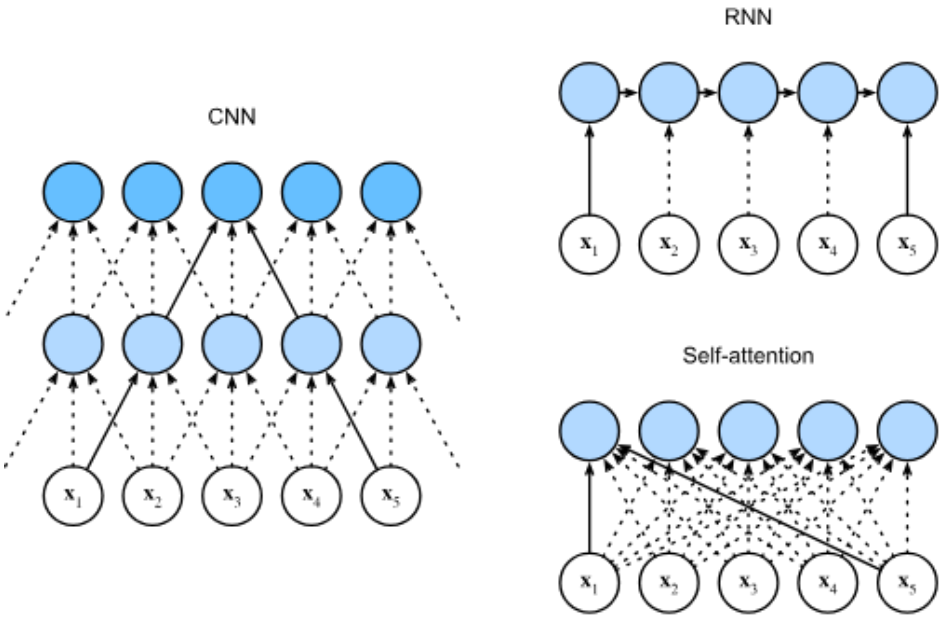

Figure 2.1: Comparing CNN, RNN, and self-attention architectures.

Self-Attention Mechanism

Self-attention computes the relevance of each word to every other word in a sequence, generating a weight matrix that reflects these relationships. Given a query Q, key K, and value V, the self-attention mechanism is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where d_k is the dimension of the key vectors. This enables the model to focus on relevant parts of the input text based on the query, which is essential for tasks like summarization.

Encoder-Decoder Structure

The T5 model, like many other summarization models, uses an encoder-decoder architecture. The encoder transforms the input text into a series of hidden states representing contextualized word embeddings. The decoder, then, generates output tokens based on these embeddings, facilitating the generation of meaningful summaries.

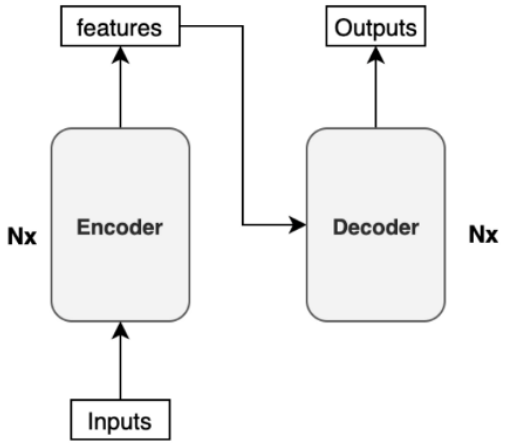

Figure 2.2: Encoder Decoder architecture

2.2.2 T5-Specific Techniques

Text-to-Text Framework

The T5 model is unique in framing every NLP task, including summarization, as a text-to-text problem. This approach simplifies training by using a single model for multiple tasks. For summarization, the input text is encoded, and the model generates a summarized version as output text.

2.2.3 Evaluation Metrics

ROUGE

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric is commonly used to evaluate summarization models. ROUGE measures the overlap of n-grams, word sequences, and word pairs between the generated summaries and reference summaries. Key ROUGE metrics include ROUGE-1, ROUGE-2, and ROUGE-L, which represent unigram, bigram, and longest common subsequence overlap, respectively.

BLEU

BLEU(Bilingual Evaluation Understudy) is another metric for evaluating text generation, measuring the overlap of words and phrases. Although it was originally designed for machine translation, BLEU has been adapted to assess the fluency and accuracy of summarization models.

Human Evaluation

In addition to automated metrics, human evaluation is crucial for assessing the quality of generated summaries, particularly for subjective criteria such as readability, coherence, and informativeness. Human evaluation provides a qualitative measure that complements quantitative metrics like ROUGE and BLEU.

By leveraging state-of-the-art transformer models, large datasets, and suitable evaluation metrics, our project aims to address the challenges of multilingual summarization for personalized news delivery.

3. Methodology

3.1 Data Collection Methodology

For this project, the primary dataset comprises a mix of international and local news data to ensure comprehensive coverage across languages and topics. The collection process involved two main data sources:

CNN-DailyMail Dataset

The *CNN-DailyMail News Text Summarization* dataset, obtained from Kaggle, serves as a robust English-language dataset for summarization tasks. This dataset includes approximately 311,971 news articles, each paired with human-written highlights, making it suitable for training and testing the model. This dataset isn't used for finetuning, and only to train the custom T5 model.

Web Scraping from Ekantipur Website

To gather Nepali-language data, we scraped news articles from the *Ekantipur* website, a leading news portal in Nepal. This involved collecting articles from various categories to ensure diversity and relevance. For this, we used Python's `BeautifulSoup` and `requests` modules, which enabled efficient and automated extraction of news content.

3.2 Data synthesis

We prepared the summaries of the collected news articles using manual summarization and tools like ChatGPT and Gemini.

3.3 Data preprocessing

Tokenization

We tokenized the news articles and the corresponding summaries for further processing so that the english words can be interpreted for machine understandability. The tokens were then encoded using Positional encoding.

We used T5Tokenizer from Hugging Face which is a Subword-based Tokenization (Byte-Pair Encoding - BPE), for the purpose of Fine Tuning the model. The T5Tokenizer comes with a pre-built vocabulary, shared between encoder and decoder, based on the T5 model's training corpus (e.g., C4 dataset).

Also, we used a Word-level Tokenization which has a word based vocabulary. The Tokenizer class builds a vocabulary from the training data by splitting text into words. Each unique word is assigned a numerical token ID, based on its frequency in the corpus.

Positional encoding

The positional encoding in this project captures the order of words in a sequence, which is essential for sequence-based models like Transformers that lack inherent recurrence or convolution. It creates unique embeddings for each position in the sequence by combining sine and cosine functions of varying frequencies, allowing the model to differentiate between word positions. This method effectively provides the model with relative and absolute position information, crucial for context understanding in tasks like summarization.

3.4 System Architecture

The system follows a **three-tier architecture**, comprising the **client**, **server**, and **database** tiers. This architecture supports the separation of concerns and enhances scalability and maintainability.

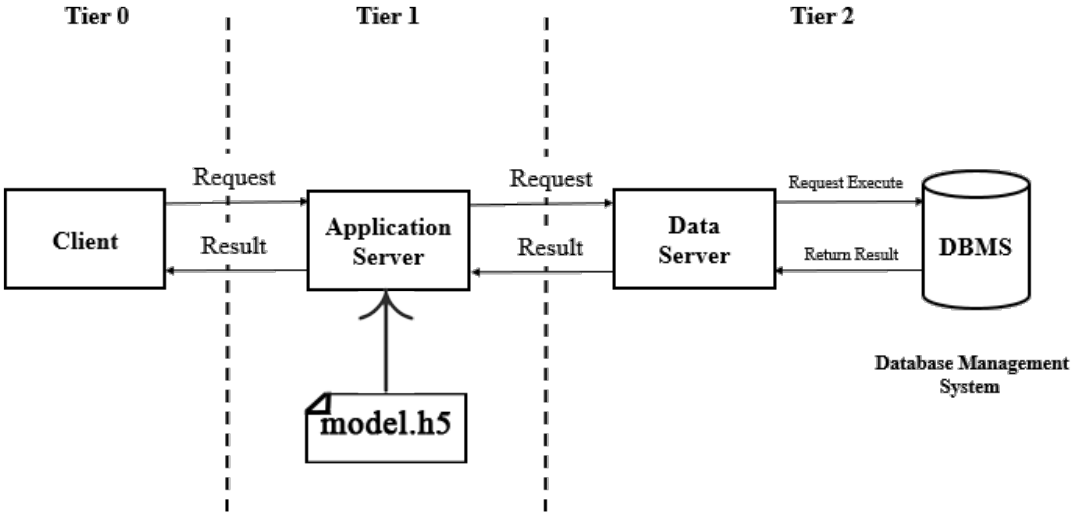

Figure 3.1: System Architecture

3.5 Transformer Model and T5 Implementation

We developed a Deep Learning model inspired from T5 architecture for summarization, which treats summarization as a sequence-to-sequence task. The T5 model's *encoder-decoder architecture* and self-attention mechanism make it well-suited for handling lengthy news articles and generating coherent summaries.

We used the CNN-DailyMail News Text Summarization, and our own dataset concatenated

for training the custom model. But, due to less amount of resources we weren't able to complete it as expected. Hence, it has been trained, its working, but the results are quite dull.

3.6 Transfer learning

Due to resource-timing constraints and very steep learning curve, we could not accomplish the task of implementing the T5 model from scratch. Rather, we used a pretrained T5 model and fine tuned it over our data and achieved successful results. We used our own scraped labeled data to fine tune the pre trained model.

3.7 Infrastructure

To train and fine tune the model, we relied on cloud-based resources, such as **Google Colab** and **Kaggle**, which provide free GPU support. This allowed for efficient model training without the need for high-end local hardware.

T5 For Conditional Generation (t5-small)

T5-small is a lightweight version, making it suitable for fine-tuning on smaller datasets and for environments with limited computational resources. We used the following variables during the training and the setup:
Learning rate=2×10^{-5}
$Parameters size = 60M$
$Per device train batch size = 8$
$Per device eval batch size = 8$
$Epochs = 100$
$weight decay = 0.01 : Regularization to prevent over fitting by penalizing large weights.$
$eval strategy =' epoch' : Model is evaluated at the end of every epoch$

The tokenizer truncates or pads inputs and outputs such that **max length is 1024** for inputs and 150 that for outputs. Preprocessing was automatically handled by the datasets library.

Input Formatting: **"summarize: article"** as input helps the model understand the task.

The loss curve was an almost perfect decaying exponential graph, with
Last epoch loss of 1.4073741436004639
Training loss of 0.955800.

3.8 Frontend and Backend Development

Frontend and backend components were actively developed using *Visual Studio Code* as the main Integrated Development Environment (IDE). Each component was implemented with specialized languages and tools to enhance functionality and maintainability.

3.8.1 Backend Development

The backend of the application was developed using **Django** and the **Django Rest Framework (DRF)** in Python, enabling a modular and RESTful approach. The backend was divided into distinct applications, such as *account*, *api*, *crawler*, and *news*, among others. This modularization followed Object-Oriented Programming principles, enhancing the flexibility and scalability of the backend system.

Database Integration

The backend supports multiple database options, such as MySQL, SQLite, and PostgreSQL. During the development phase, a lightweight database (sqlite3) was used, with plans to switch to PostgreSQL in production to handle larger volumes of data more efficiently.

3.8.2 Frontend Development

The frontend of the application was built as a cross-platform mobile application using Flutter, ensuring it works seamlessly on multiple platforms for greater accessibility. Key features and tools used include:

- **Firebase**: Used for backend integration, user authentication, and real-time data management.
- **Lottie**: Used to create engaging animations and enhance the user interface.
- **flutter_secure_storage**: Used to securely store sensitive user data, such as tokens and login credentials.
- **Google Authentication**: Implemented for secure user login and authorization.

All interactions between the frontend and backend were secured to protect user data and privacy.

3.8.3 Version Control and Development Process

GitHub was used for version control and to facilitate pair programming. The team adopted an Agile **Scrum** process, with regular sprint meetings to review progress, identify issues, and define upcoming tasks. This iterative approach enabled faster resolution of issues and continuous improvement.

3.8.4 Backend-Frontend Integration

The backend was deployed using the free-tier plan on PythonAnywhere. The APIs were integrated with the frontend mobile application and rigorously tested to ensure smooth functionality. The system integration was executed successfully, achieving seamless interaction between the backend and frontend components.

3.8.5 Future Training and Optimization

As we progress, we aim to continue utilizing Google Colab and Kaggle for model training and optimization. This approach will allow for the continuous refinement of the T5 model until it meets the performance requirements necessary for deployment.

3.9 System Testing

Extensive testing was performed on both the frontend and backend components to ensure stability and performance. Functional and integration testing were carried out regularly as part of the Scrum sprints.

4. Experimental Setup

4.1 Dataset preparation

Data collection was a fundamental part of the project, as it provides the basis for training the T5-based summarization model. Two datasets were obtained to cover both international and local news content:

4.1.1 CNN-DailyMail Dataset

The *CNN-DailyMail News Text Summarization* dataset was sourced from Kaggle. This dataset, amounting to approximately 1.27 GB in size and containing 311,971 entries, includes a variety of news articles in English along with their summaries, labeled as *highlights*. The dataset files were organized for training, validation, and testing to facilitate efficient model training and evaluation.

4.1.2 Ekantipur News Scraping

To build a dataset for Nepali news, we scraped articles from the Ekantipur website using Python's `BeautifulSoup` and `requests` modules. News articles were collected from the following categories:

- News
- Business
- Opinion
- Sports
- National
- Entertainment
- Photo Feature
- Feature
- World
- Blog
- Education
- Health

Data Fields The scraped data included fields such as:

- **Article Content:** Full text of the news article.
- **Summary:** Short, human-written summary, labeled manually or with tools.
- **Category:** The category to which the news item belongs.
- **URL:** The link to the original article for reference.
- **Author:** Author information, where available.
- **Image URL:** Link to any associated image (for future use).

Each category includes 100 articles, resulting in a balanced and diverse dataset. The scraped data includes essential fields such as article content, manually created summary, category, URL, author information, and image URL. For certain articles, manual labeling was done to generate the summaries, with support from tools like ChatGPT where necessary. This dataset is now available for model training and testing, providing both linguistic and contextual diversity for the project.

4.2 Tech Stack

4.2.1 Data Science

In this project, we focused extensively on Deep Learning, dedicating significant time to learning and utilizing TensorFlow/Keras. These frameworks were heavily used to train the custom T5 model from scratch. For fine-tuning the model, we leveraged Hugging Face's `transformers` library along with PyTorch, which provided an efficient and robust platform for this purpose.

In addition to the core frameworks, we employed several auxiliary libraries, such as `numpy`, `pandas`, `datasets`, `evaluate`, and `pickle`, to handle various tasks as required throughout the development process.

4.2.2 Frontend

The frontend was developed as a mobile application using **Flutter**. This cross-platform app provides a user-friendly interface for accessing personalized news summaries. Major features include:

- **Login and Google Authentication:** Secure login and authentication were managed via the backend, allowing users to register and log in through Google.
- **News Feed and Summary Display:** The app displays a personalized news feed with summaries, links to full articles, and options to read summaries aloud.

4.2.3 Backend

The backend, developed using **Django** and **Django Rest Framework (DRF)**, has been fully implemented. Major components of the backend include:

- **API Endpoints:** RESTful APIs were created for user authentication, news retrieval, and news summary generation. Each API endpoint was rigorously tested to ensure performance and reliability.
- **Database Integration:** The backend is such that supports multiple types (e.g., MySQL, SQLite, PostgreSQL). For development purposes, SQLite is being used, with plans to transition to PostgreSQL in production for better performance and scalability.

4.3 Two-Model Approach

We developed two models using two different approaches. The first model was built entirely from scratch, where we designed its architecture and implemented all preprocessing steps manually. While the model was successfully implemented, it faced challenges with output quality. It struggled to understand context effectively and frequently introduced out-of-vocabulary tokens. Additionally, the limited dataset and computational resources hindered its performance.

As a result, we shifted to fine-tuning a pre-trained T5 transformer model from Hugging Face. This approach allowed us to overcome the previous limitations. We fine-tuned the model, performed all necessary adjustments, and achieved significantly better results. The following sections provide a detailed discussion of the fine-tuned model's performance and metrics.

5. System Design

The diagram represents a pipeline for generating audio news summaries. News is collected from various sources, scraped, and preprocessed to produce structured data. A T5 Transformer model is fine-tuned to summarize the news, which is then integrated into an app. The app converts the summaries into audio using text-to-speech technology, delivering them as user-friendly news bulletins. This system streamlines the process of transforming raw news into concise, accessible audio formats.

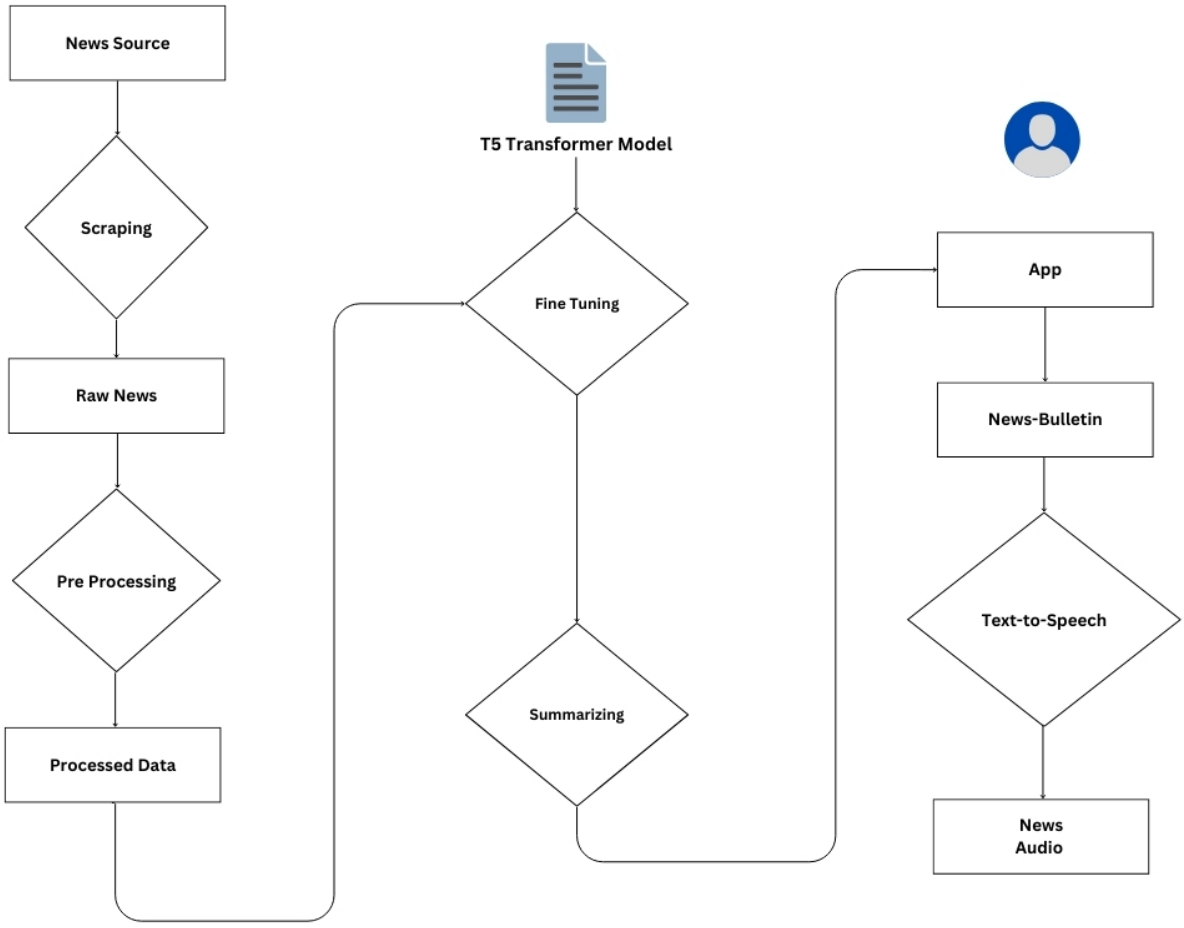

Figure 5.1: System block diagram

6. Results and Discussion

6.1 Evaluation metrics

Training and Validation Loss

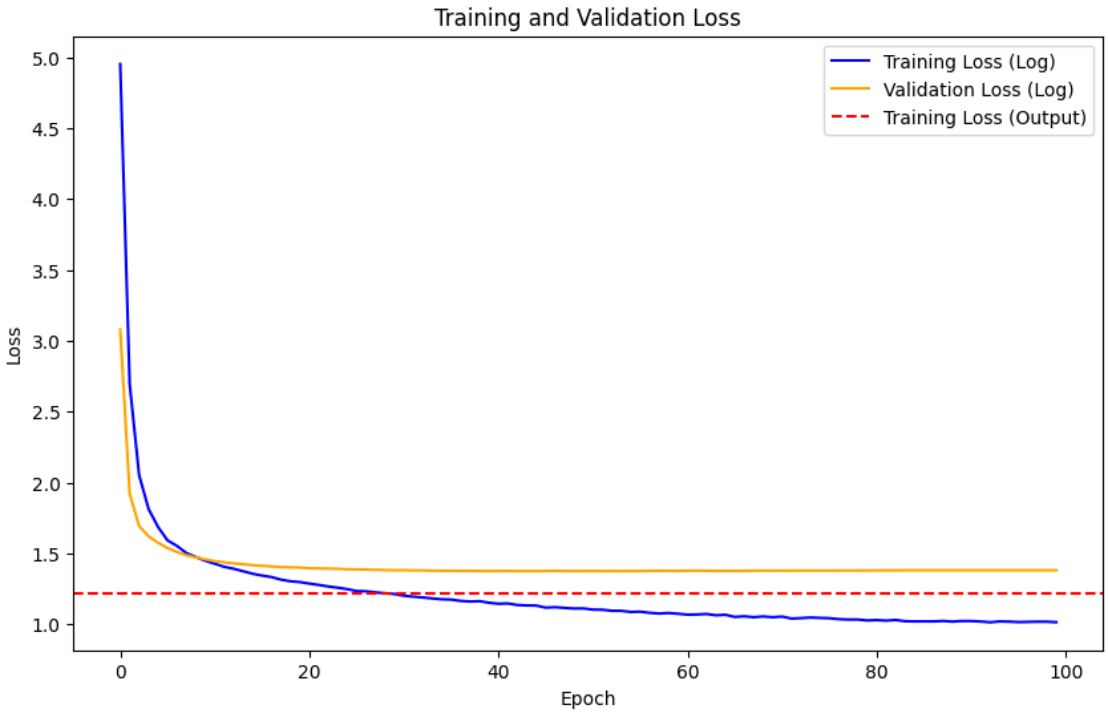

Figure 6.1: Training Loss vs Validation Loss graph for 100 epoch

This figure shows the training and validation loss trends over 100 epochs. Both losses decrease, indicating effective model training and generalization. The red dashed line represents a constant benchmark or output layer loss. The plot confirms steady optimization without signs of overfitting.

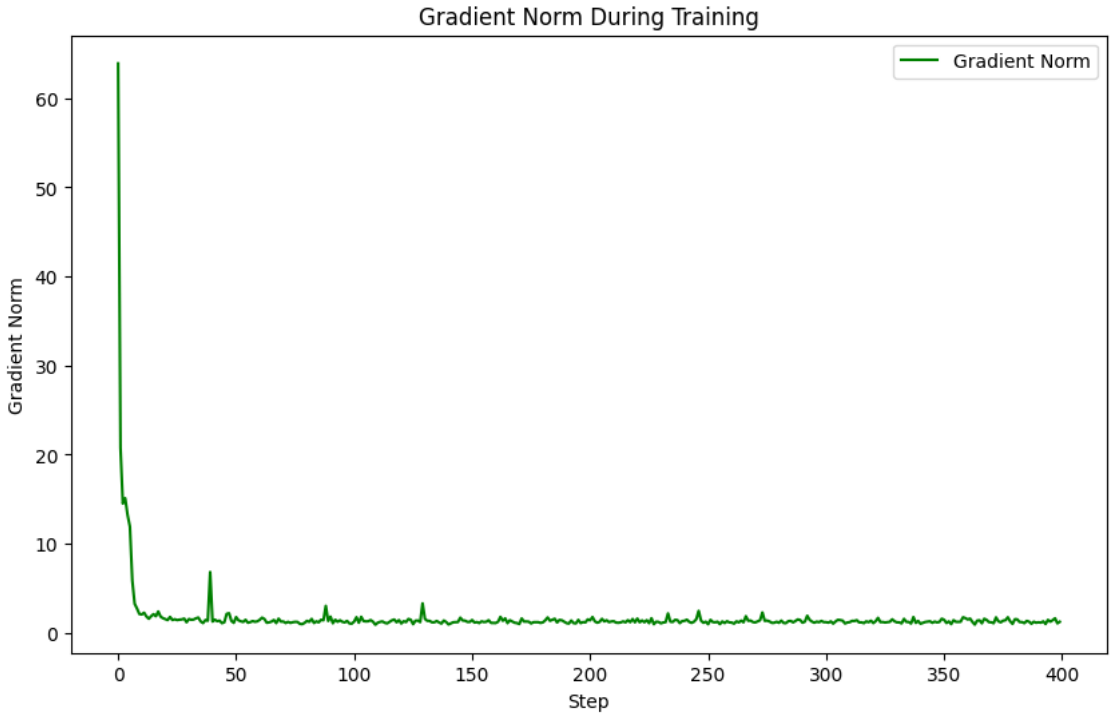

Figure 6.2: Gradient normalized for 100 epochs (400 x 10 steps)

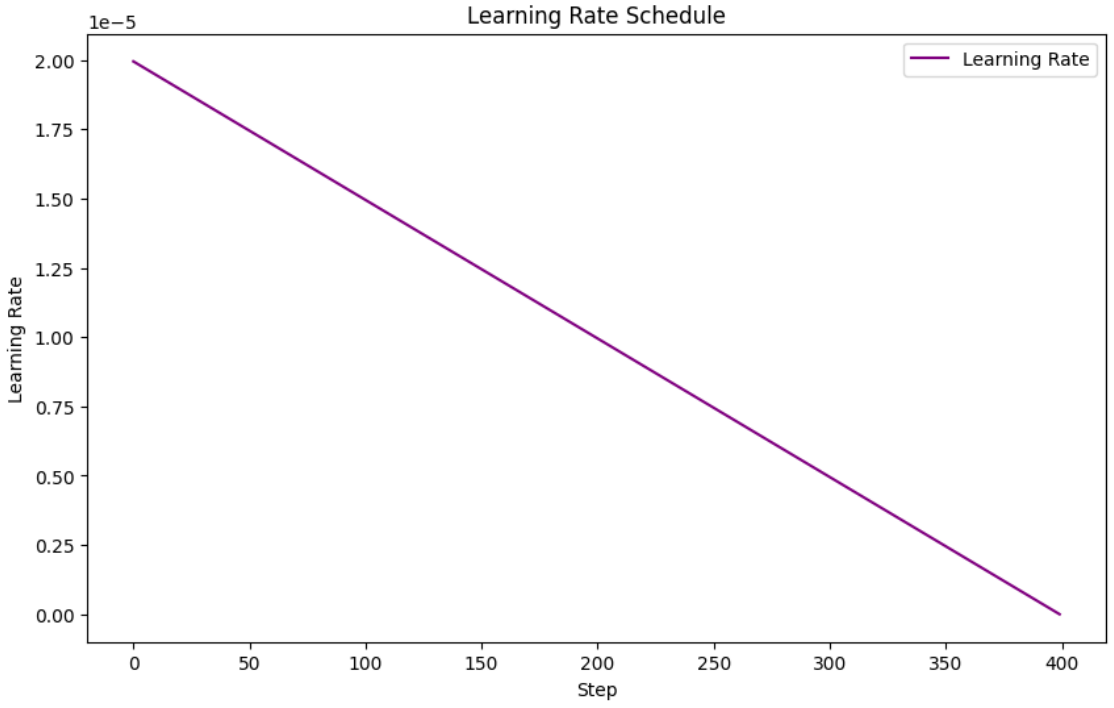

Figure 6.3: Learning rate schedule during training

6.2 Model Performance Comparision

6.2.1 On Seen Data

Original Summary from the dataset:

New Zealand's historic 3-0 Test series whitewash over India has shocked the cricket world. The Black Caps won the final Test by 25 runs in just three days, marking India's second-ever home clean sweep loss. Darrell Mitchell praised his team's unity, while William Young, who scored 244 runs, was named 'Player of the Series.' Indian captain Rohit Sharma took responsibility for the poor performance, acknowledging the tough defeat and facing criticism from former players.

Summary from the pre trained model:

new Zealand batsman darrell Mitchell believes his team is made up of a whole Kiwi. the same team created a historic 3-0 whitewash in the Test cricket series against india. the black caps won the first Test in Bengaluru for the first time since 1988.

Summary from the fine tuned model:

New Zealand batsman Darrell Mitchell believes his team is made up of a whole Kiwi. The Black Caps won the first Test in Bengaluru for the first time since 1988, with Kane Williamson recovering from injury, and his only aim is to prepare for the series against England. The loss was the biggest in New Zealand cricket history, with Tim Southee stepping down after a 2-0 defeat in Sri Lanka. The team's performance was a 'clean sweep', with Indian captain Rohit Sharma claiming responsibility for the loss. Mitchell says, 'I am very saddened by the loss.

6.2.2 On Unseen Data

Summary from the pre trained model:

government's top priority is how to conduct agile, sound, competent and transparent administration. he said that while protecting the forest environment, development projects should be facilitated in this area and forest-related agencies should not work in such a way as to hinder development. he drew the attention of ministers and secretaries that such wrong trends should not happen.

Summary from the fine tuned model:

Prime Minister KP Sharma Oli addressed public complaints about governance reforms at the 53rd meeting of the National Development Problem Solution Committee in Singhdarbar, on

Friday. He emphasized the need for agile, sound, competent and transparent administration. He stressed that forest-related agencies should not work in such a way as to hinder development. He emphasized the importance of modernizing agriculture and encouraging private sector investment. He stressed that social chaos has been caused to hinder development.

6.2.3 ROUGE Metric

We have measured the ROUGE metric for better understanding of the model before and after finetuning, and we get:

Metric	**t5-small**	**t5-small fine tuned**
Rouge 1	0.3902900937813921	0.5362305568681147
Rouge 2	0.1736686323908947	0.26642750147837807
Rouge L	0.27117293849375756	0.37021276378138457
Rouge L Sum	0.27235231165381946	0.37028279038223355

Table 6.1: Performance comparison between t5-small and t5-small fine-tuned models based on Rouge metrics.

The results in Table 6.1 clearly demonstrate that our fine-tuned T5-small model significantly outperforms the base T5-small model across all ROUGE metrics. This highlights the effectiveness of fine-tuning in improving the model's ability to generate high-quality and contextually accurate summaries.

7. Future Enhancements

7.1 Model Performance Enhancement

To further enhance the performance of the model, future efforts will focus on fine-tuning it with more diverse and larger datasets. In particular, we plan to incorporate datasets from *Annapurna Post* and other credible sources. Expanding the training data will help the model generalize better, improving its accuracy and versatility. Additionally, efforts will be made to build more English language models, while simultaneously developing separate models for Nepali datasets and summaries. Subsequently, the project will also aim to create summarization models for other local languages such as Newari and Maithili, enabling broader inclusivity.

7.2 Deployment

The deployment phase marks the transition from development to production. Several steps are planned to ensure successful deployment:

7.2.1 Backend and Model Deployment

- **Server Setup:** The backend and trained model will be hosted on a scalable and reliable infrastructure, such as *AWS EC2* or *SageMaker*. These platforms provide the necessary computational resources and flexibility for handling API requests efficiently.
- **Database Configuration:** The system will be integrated with a production-ready database, such as *PostgreSQL*, to support a larger user base and offer robust, reliable data storage.

7.2.2 Mobile Application Deployment

- **Publishing on Google Play Store:** The Flutter-based mobile application will be published on the Google Play Store, ensuring ease of access for end-users.
- **User Support and Feedback Mechanism:** A feedback mechanism will be incorporated to collect user reviews and insights, which will help in identifying areas for future improvement and debugging.

Successful deployment will enable users to interact with the application seamlessly, offering personalized news summaries directly on their mobile devices.

7.3 Additional Features

7.3.1 Multilingual Capabilities

To cater to a broader audience, the system will integrate multilingual functionality. Users will be able to select their preferred language, allowing the application to generate news summaries in Nepali, Newari, Maithili, and other local languages. This feature will promote inclusivity and make the application accessible to a diverse user base.

7.3.2 Credibility Verification

Incorporating a feature to verify news credibility by cross-referencing multiple news sources is a planned enhancement. This will help ensure that the summaries provided are accurate, reliable, and trustworthy.

7.4 Exploration of Custom Text-to-Speech (TTS) Models

As part of learning and experimenting with transformers, future work will explore building custom text-to-speech (TTS) models tailored to this project. This feature would allow the application to read out news summaries to users, enhancing accessibility for individuals with visual impairments or those who prefer audio content.

7.5 Scaling for Larger Datasets

To address memory and computational challenges, advanced optimization techniques, such as gradient accumulation, mixed precision training, and efficient dataset loaders, will be implemented. These measures will ensure that larger datasets can be processed without overwhelming available resources.

8. Conclusion

To conclude, the project has successfully delivered a personalized news summarization application powered by a custom fine-tuned T5 transformer model. Through the systematic integration of collected datasets, advanced model architecture, and seamless backend-frontend connectivity, the application provides concise, relevant, and personalized news summaries. Key achievements include the successful fine-tuning of the T5 model on a robust dataset of CNN-DailyMail articles and curated Nepali news, deployment of a modular Django backend, and a fully functional Flutter-based mobile app that integrates user authentication and personalization features.

The application is now live and accessible, offering a smooth user experience and the ability to adapt summaries to user preferences. By delivering concise and tailored news summaries, the application empowers users to stay informed efficiently. As the project transitions to its public deployment phase, we are confident in its potential to navigate the overwhelming information landscape and set a foundation for future advancements in AI-driven summarization tools.

Bibliography

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[2] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

9. Appendices

9.1 Mobile Application Interface

9.1.1 Login Screens

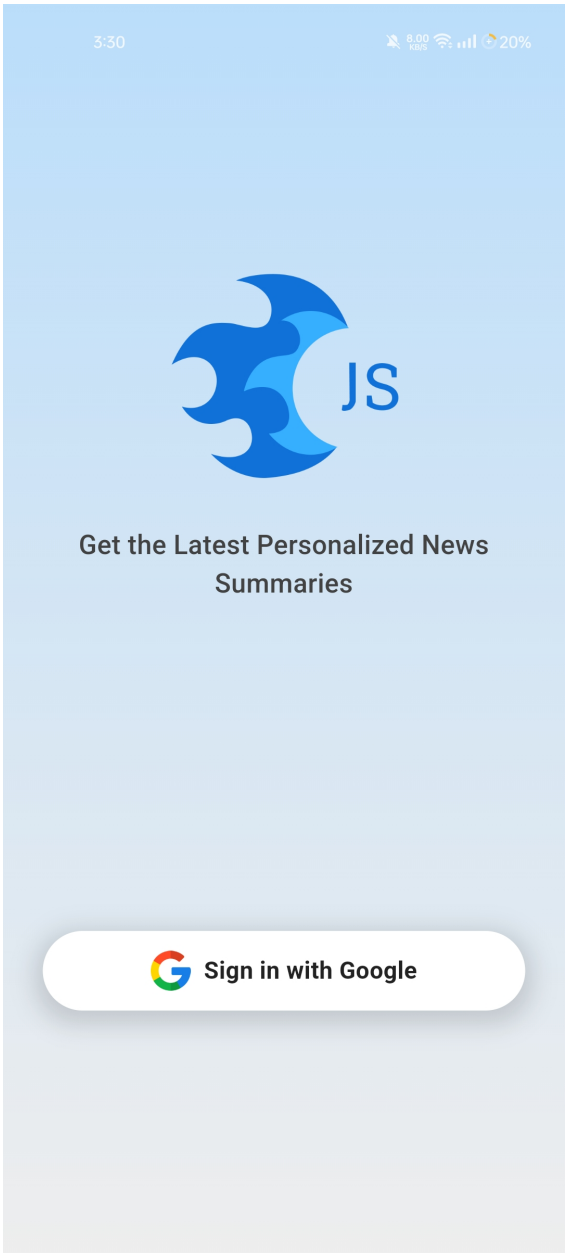

Figure 9.1: Login Screen

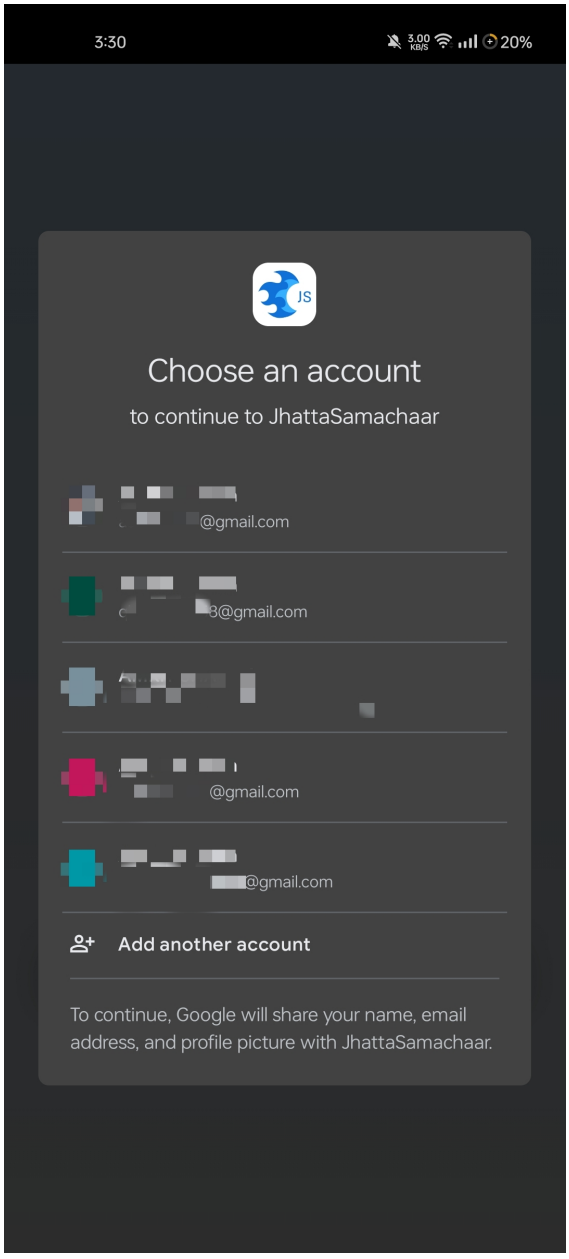

Figure 9.2: User Selection Screen

9.1.2 Homepage Screens

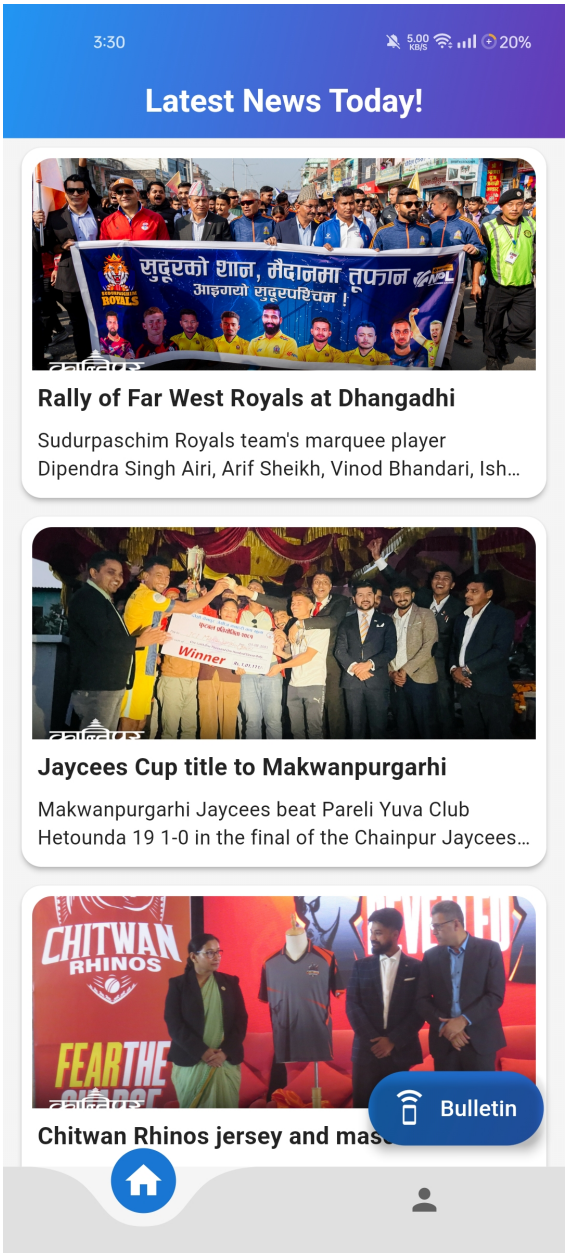

Figure 9.3: Homepage

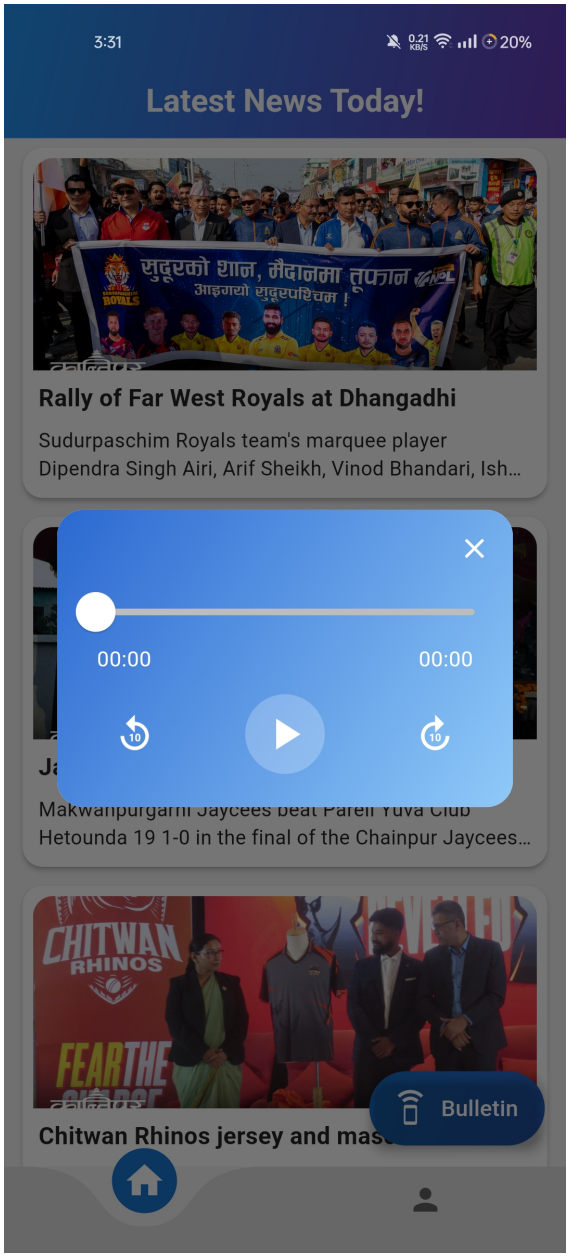

Figure 9.4: Audio Player Interface

9.1.3 Account Screens

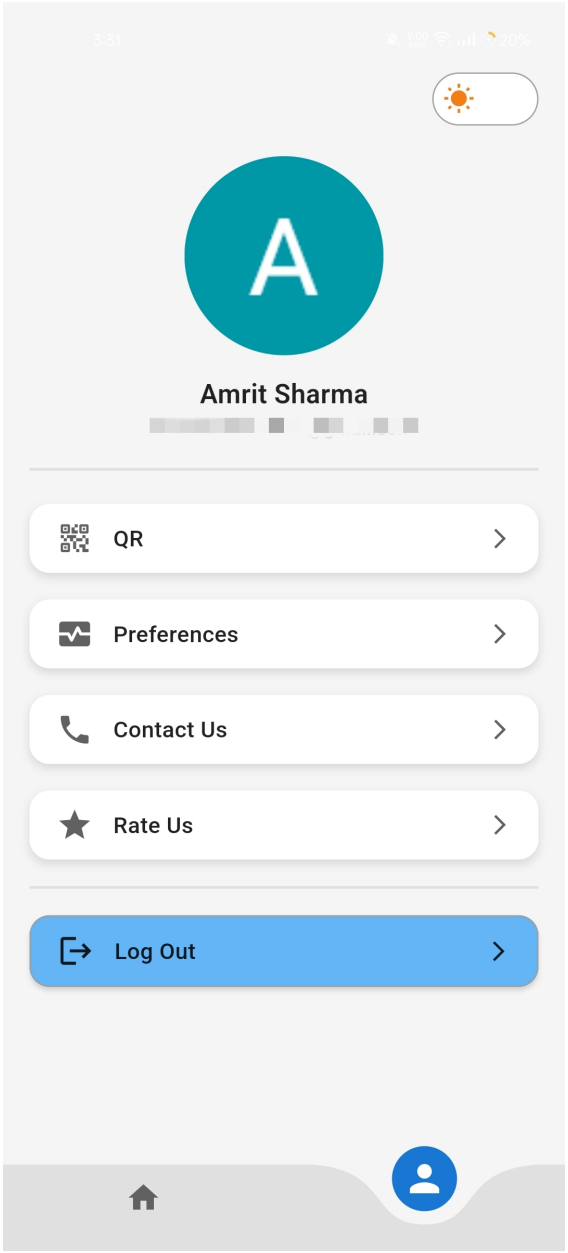

Figure 9.5: Account Page

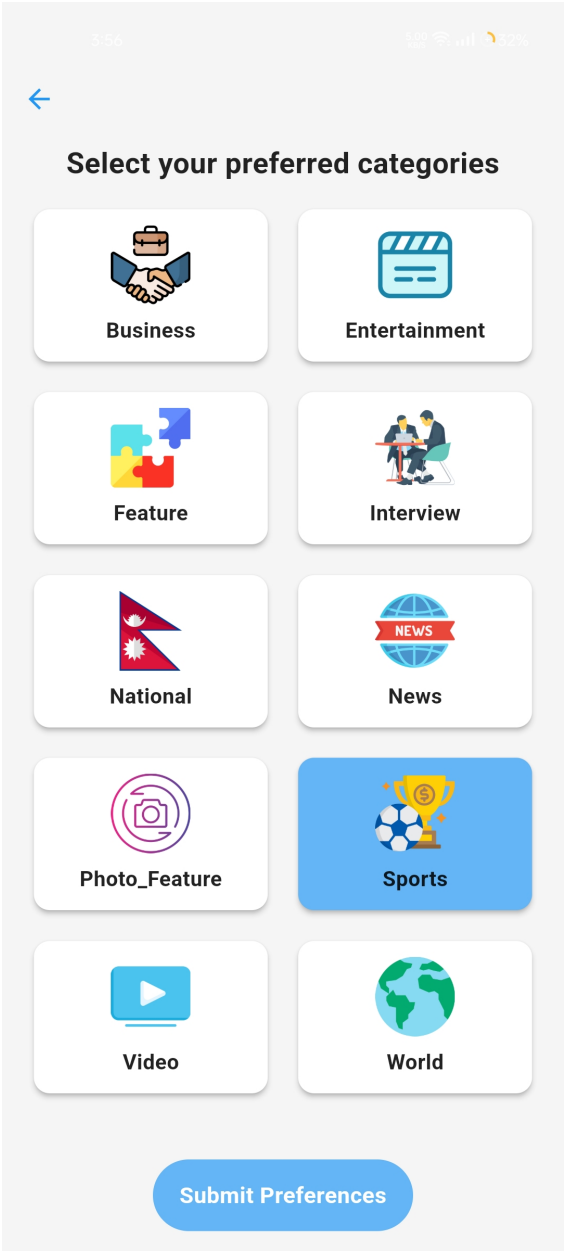

Figure 9.6: Preference Selection Page