TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A

MID TERM STATUS 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

Abstract

This report presents the development progress of a personalized news summarization application that leverages a custom-built T5 transformer model to generate concise, user-specific news summaries. Using a comprehensive dataset combining the CNN-DailyMail corpus and manually labeled Nepali news articles scraped from Ekantipur, we aim to create culturally relevant summaries tailored to user interests. The project's backend, developed with Django, and the Flutter-based mobile frontend have been fully integrated and tested locally, providing seamless functionality. Remaining tasks include finalizing the T5 model implementation, integrating it with the backend, and deploying the full application on the Google Play Store. This work is an essential step toward a personalized news experience, helping users access important information quickly and efficiently.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

The growth of digital media has led to an overwhelming amount of news data online. Users often struggle to keep up with relevant information amidst this vast flow. News summarization, where essential highlights are extracted, can greatly improve user experience by delivering concise, personalized news summaries. In light of these needs, our project leverages the power of applied data science, particularly in natural language processing (NLP) with transformer-based models, to deliver meaningful, tailored news summaries. With a focus on both English and Nepali news data, our project utilizes both well-established datasets and real-time scraped content, enabling a comprehensive approach to personalized news delivery.

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 a T5 transformer model from scratch 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 "CNN-DailyMail News Text Summarization" dataset and scraped news data from Ekantipur for supervised training.

2. Model Development: Constructing a transformer-based T5 summarization model capable of handling multilingual content.

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

4. Deployment and Commercialization: Once the model is built and the system integrated, the final stage will focus on deploying the application and backend in production for user accessibility.

The project has achieved significant milestones, with the dataset collected and processed, backend and frontend developed and integrated, and a clear roadmap for the remaining work on model building, system integration, and deployment. This comprehensive progress sets the foundation for meeting the project's objectives and delivering an effective news summarization solution.

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 like CNN-DailyMail.

2.2 Related Theory

This section delves into the theoretical background supporting the techniques and models used in the project, focusing on transformer models, encoder-decoder architecture, and summarization evaluation metrics.

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.

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.

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.

In summary, this chapter has outlined the key methodologies and theoretical foundations for the project. 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:

3.1.1 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.

3.1.2 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 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.

3.2.1 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.

3.2.2 Frontend Development

The frontend was designed as a cross-platform mobile application using **Flutter**. Flutter's compatibility allows the application to be deployed on multiple platforms, maximizing accessibility. All frontend-backend interactions were secured, with **Google Authentication** implemented for user login and authorization.

3.2.3 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.

3.2.4 Backend-Frontend Integration

Integration between the backend and frontend was achieved using *Microsoft's port forwarding feature* in Visual Studio Code, allowing the local backend to be accessible over the internet for testing purposes. This temporary port forwarding feature enabled real-time testing on mobile devices, with regular application testing conducted to ensure seamless communication between components.

3.3 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.

3.4 Model Training

3.4.1 Infrastructure

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

3.4.2 Transformer Model and T5 Implementation

The project uses a **T5-based transformer model** for text 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. Initial model implementation has been conducted, currently paused due to programming language-level error, with further tuning and optimization planned as the data collection completes.

3.4.3 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.

3.5 Testing and Deployment

3.5.1 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.

3.5.2 Deployment Plans

Future deployment plans include hosting the backend on a cloud platform to make it publicly accessible, finalizing the frontend for distribution on the Google Play Store, and implementing the trained T5 model to deliver news summaries. This final deployment phase will signify the project's transition from development to production.

In summary, the proposed methodology involves a structured approach for data collection, system development, and model training. With a focus on transformer models, large datasets, and rigorous testing, this methodology aims to create a robust platform for personalized news summarization.

4. Tasks Completed

This chapter outlines the major tasks completed so far in the project, focusing on data collection, backend and frontend development, system integration, and research on transformer models, particularly T5.

4.1 Data Collection

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 Backend Development

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.
- **Modular Architecture:** The backend was divided into modules such as `account`, `api`, `crawler`, `models`, and `news`. This modular structure simplifies maintenance and scalability.

- **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.

The backend is now fully integrated with the frontend and provides seamless data access and processing capabilities, with Google Authentication incorporated for secure user login.

4.3 Frontend Development

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 are 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.
- **Cross-Platform Compatibility:** Flutter was chosen to ensure that the app can be deployed on multiple platforms, enhancing its accessibility.

The frontend was tested thoroughly to ensure compatibility and responsiveness, completing the user interface for the project.

4.4 System Integration

The integration of the backend and frontend was completed and tested locally. Using *Microsoft's port forwarding feature* in Visual Studio Code, the local backend was made accessible over the internet, allowing the mobile app to interact with it seamlessly.

4.4.1 Version Control and Collaboration

GitHub was used for version control, enabling team members to collaborate effectively and manage changes systematically. The team adopted a Scrum methodology with regular sprint meetings, facilitating iterative development and feedback cycles.

4.4.2 Testing

Integration testing was conducted to confirm the backend and frontend were functioning as intended together. Minor issues were resolved, and the integration was verified to work seamlessly in a local environment. The integration ensures real-time data flow and smooth user interaction within the app.

4.5 Study of Transformer Models and T5 Architecture

The T5 model was chosen for its versatility in text summarization tasks. Research and initial trials were conducted to gain a deep understanding of transformer architectures and T5 specifics, including:

- **Transformer Architecture:** We studied the encoder-decoder structure, multi-head attention, and positional embeddings, which are crucial for handling sequential text data effectively.
- **T5 Model Structure:** The team focused on understanding the T5 model's pre-training objectives, particularly the *text-to-text* framework, which is well-suited for summarization. We explored the nuances of self-attention mechanisms, feed-forward layers, and learned embeddings specific to T5.
- **Experimentation and Testing:** Initial tests were conducted to implement a simplified transformer model. Using resources like Google Colab and Kaggle, we familiarized ourselves with the training process and optimized the model architecture for news summarization tasks.

This foundational understanding of the T5 model prepares us for the next stage, where we will implement and train our custom T5 transformer from scratch, leveraging both local resources and cloud platforms.

4.6 Summary of Completed Tasks

Significant progress has been achieved, including:

- Successful data collection from both international (CNN-DailyMail) and local (Ekantipur) sources.
- Full implementation of the Django backend, with modular structure and database integration.
- Complete development of the Flutter-based frontend, providing a cross-platform, user-friendly interface.
- Comprehensive integration of the backend and frontend, with extensive testing for smooth operation.
- A strong foundational study and initial experimentation with the T5 transformer model, positioning us well for custom implementation and training in the upcoming stages.

With these tasks completed, the project is well-prepared to proceed with model training, final integration, and deployment.

5. Tasks Remaining

This chapter outlines the key tasks remaining for the completion of the project. The major tasks include building the custom transformer model, integrating the entire system, and deploying the application for user access.

5.1 Model Development

The primary technical task left is the development and training of the T5 transformer model. The model will be built from scratch to meet the project's specific requirements for personalized news summarization.

5.1.1 Custom T5 Transformer Model

The main objective is to implement a custom T5 transformer model, which involves:

- **Architecture Design:** Writing the architecture for the T5 model based on the transformer encoder-decoder design. This includes defining layers, self-attention mechanisms, and training objectives specific to our summarization needs.
- **Debugging and Optimization:** As the model is being implemented from scratch, extensive debugging and performance optimization will be necessary to ensure efficient and accurate text summarization.
- **Training:** The training process will leverage both the CNN-DailyMail and Ekantipur datasets, using GPU resources from platforms like *Google Colab* and *Kaggle* to expedite processing. This training phase will involve multiple iterations to fine-tune the model for optimal performance.

This task represents a critical milestone, as the custom model will be central to the project's functionality, enabling high-quality personalized news summaries.

5.2 System Integration

Once the custom T5 model is successfully trained and tested, it will be integrated with the existing backend and frontend. This integration involves:

- **Model-Backend Integration:** Deploying the trained model as a service within the backend to allow real-time summarization requests from the mobile application.

- **Testing and Validation:** Comprehensive testing to ensure the model's outputs are accurate, relevant, and correctly formatted for the frontend display. This phase will also involve validation with both international and Nepali news data to evaluate model performance across diverse content.

Successful integration will allow the complete system to operate seamlessly, with backend processing, model inference, and frontend display working in sync.

5.3 Deployment

The final phase of the project is the deployment of the full application. Deployment tasks include:

5.3.1 Backend and Model Deployment

- **Server Setup:** The backend and trained model will be deployed on a server capable of handling API requests, preferably using a scalable and reliable service such as *AWS* or *Google Cloud Platform*.

- **Database Configuration:** Migration to a production-ready database such as PostgreSQL to support a larger user base and provide reliable data storage.

5.3.2 Mobile Application Deployment

- **Publishing on Google Play Store:** The Flutter mobile application will be published on the Google Play Store, making it available for user download and interaction.

- **User Support and Feedback Mechanism:** Setting up a mechanism for users to provide feedback, which will be valuable for future improvements and debugging.

Deployment will mark the commercial release of the project, allowing users to access personalized news summaries on their mobile devices.

5.4 Summary of Remaining Tasks

In summary, the remaining tasks include:

- Development and training of the custom T5 transformer model for news summarization.

- Full system integration, connecting the model with the backend and frontend for seamless user experience.

- Deployment of the backend, model, and mobile application, with publication on the Google Play Store and final production-ready configurations.

These tasks are critical to completing the project, enabling its full functionality, and preparing it for public release.

6. Conclusion

This project aimed to develop a personalized news summarization application, integrating a custom T5 transformer model to generate concise news summaries. Throughout this journey, significant progress has been made in data collection, backend and frontend development, integration, and preliminary work on the model architecture.

The project has made significant progress by collecting a robust dataset, combining CNN-DailyMail data and manually curated Nepali news from Ekantipur, to enhance summarization relevance and quality. Backend development, achieved through a modular Django Rest Framework setup, and the complete, Flutter-based mobile app frontend have been successfully integrated, supporting features like Google authentication and personalized news with a smooth user experience. Key tasks still ahead include developing a custom T5 transformer model, fully integrating it with the backend, and deploying the entire system, making it accessible to users and primed for future enhancements.

6.1 Conclusion and Future Work

In conclusion, the project is well on its way to delivering a personalized news summarization application. Each completed milestone has built a strong foundation, and the remaining tasks focus on enhancing the model's performance and preparing the application for public release.

Looking ahead, this project has the potential to expand further by incorporating additional news sources, refining model accuracy, and possibly adding more user personalization features based on feedback. With continued effort and refinement, this application can provide a unique and efficient way for users to stay informed with summaries tailored to their preferences and interests.

6.2 Closing Remarks

In summary, our project has achieved significant progress towards building an innovative and user-friendly news summarization tool. With the final development and deployment steps, we are excited to bring this application to life, helping users navigate the vast information landscape through concise, personalized summaries.

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.