NLP Project Report

Doha Hemdan 202200701

Mariam Yasser 202200886

Sarah Elsayed 202200347

Salma Sherif 202200622

1. Introduction

1.1. Summarization Model

1.1.1. Problem Statement:

With the overwhelming volume of information available on the internet, users often struggle to quickly grasp the core ideas from lengthy articles, documents, or websites. Manually reading and summarizing large amounts of text is time-consuming and inefficient.

1.1.2. Introduction:

The Summarization Model is designed to automatically generate concise summaries from a given input, which can be either a raw text passage or the content extracted from a provided URL. By leveraging natural language processing techniques, the model identifies key points and presents them in a shortened form while retaining the essential meaning. This enables users to quickly understand the main ideas without reading through the entire source material.

1.2. Question-Answer Generation Model

1.2.1. Problem Statement:

Educational and training content often requires well-formed question-answer pairs for assessments, quizzes, and learning reinforcement. However, manually creating such pairs from articles or study material is both labor-intensive and inconsistent in quality.

1.2.2. Introduction:

The Question-Answer Generation Model is built to automate the creation of question-answer pairs from a given body of text or from content fetched via a URL. By analyzing the structure and meaning of the input, the model generates relevant questions along with their corresponding answers. This tool is useful for educators, trainers, and content creators who need to produce quiz materials or interactive learning exercises efficiently.

1.3. Context Answer Generation Model

1.3.1. Problem Statement:

Users frequently seek specific answers from lengthy documents or articles but lack tools that can accurately extract relevant information based on their queries. Traditional search methods return whole documents rather than direct answers, leading to a poor user experience.

1.3.2. Introduction:

The Contextual Question Answering Model allows users to ask questions about a specific body of text or the content of a web link and receive precise, context-aware answers. The model processes the given text or extracted content and dynamically searches for the answer based on the user's question. This system is particularly beneficial for applications such as customer support, document search, and interactive chatbots where accurate and direct answers are required.

2. Coding Standards and structure

2.1. Coding Standards

- **PEP 8 Compliance**
  All Python scripts and notebooks in the project follow PEP 8 coding style. Linting was performed using flake8, and formatting was enforced using black.
- **Docstrings and Comments**
  All modules, classes, functions, and methods include docstrings following PEP 257. Meaningful inline comments are provided, particularly in sections involving complex logic.
- **Type Hinting**
  Type hints have been added to functions and classes using Python's typing module to improve code readability and enable static analysis.
- **Imports Management**
  All import statements are organized according to PEP 8

- **Virtual Environment**
  The project environment is isolated using a virtual environment created with venv. Dependencies are listed in the requirements.txt file.

2.2. Coding structure

```
├── data/
│   ├── raw/
│   ├── processed/
├── models/
├── notebooks/
├── src/
│   ├── __init__.py
│   ├── data_processing.py
│   ├── model.py
│   ├── train.py
├── tests/
│   ├── test_data_processing.py
│   ├── test_model.py
├── README.md
└── requirements.txt
```

3. Models

3.1. Summarization Model

3.1.1. Reference paper

（https://ceur-ws.org/Vol-2718/paper28.pdf)

3.1.2. Dataset

Source: (https://www.kaggle.com/datasets/sunnysai12345/news-summary)
Columns Used:

- ctext: The full article text (used as input)
- text: The summary (used as target output)

Preprocessing:

- Selected only the text and ctext columns
- Renamed columns to summary and text
- Removed any null rows

- Split into 90% training and 10% test data using train_test_split.

3.1.3. Model Architecture

- **Base Model**: t5-base
- **Tokenizer**: T5TokenizerFast
- **Framework**: PyTorch

- **Components:**
    - **Dataset Class**: NewSummaryDataset
        - Tokenizes and formats text and summaries into fixed lengths (512 for text, 128 for summary).
        - Replaces padding token 0 with -100 for correct loss computation.
    - **Data Module**: NewsSummaryDataModule
        - Provides train, val, and test dataloaders.
        - Handles batch sizing and token length configuration.

    - **Lightning Module**: NewSummaryModel
        - Defines forward pass, training, validation, and testing steps.
        - Optimizer: AdamW with learning rate 1e-4.

3.2. Question-Answer Generation Model

3.2.1. Reference paper

3.2.2. DatasetSource: ()

3.2.3. Model Architecture

3.3. Context Answer Generation Model

3.3.1. Reference paper

(https://ojs.aaai.org/index.php/AAAI/article/view/6398)

3.3.2. Dataset

Source: (https://www.kaggle.com/datasets/thedevastator/introducing-quail-a-comprehensive-reading-compre)

Columns Used:

- context: the context of answering the question
- question: the question related to the context
- answers: choices of answers
- correct_answer_id: the index of the correct answer

Preprocessing:

Here's a brief bullet-point summary of the preprocessing steps:

- **Filtered Questions**: Kept only rows where the question ends with a `?`.
- **Fixed Answers**: Converted stringified lists in the `answers` column to Python lists.
- **Extracted Correct Answer**: Used `correct_answer_id` to select the correct answer from the list.
- **Built Prompt Format**: Combined context, question, and instruction into a structured prompt ending with the correct answer.
- **Train-Validation Split**: Split the cleaned DataFrame into 80% training and 20% validation sets.
- **Converted to HF Dataset**: Transformed pandas DataFrames into HuggingFace `Dataset` objects.
- **Loaded Tokenizer**: Used Mistral tokenizer; set EOS token as padding token for causal LM.
- **Tokenized Data**: Tokenized prompts with max length 256, used input IDs as labels for causal language modeling.
- **Final Format**: Set datasets to PyTorch format for training.

3.3.3. Model Architecture

- **Base Model**: `mistralai/Mistral-7B-Instruct-v0.1`
- **Tokenizer**: `MistralTokenizer`
- **Framework:**

- **Components:**
    - **Data Processing**
        - Filtered QUAIL dataset to retain only relevant columns: *context*, *question*, *answers*, and *correct_answer_id*.
        - Removed questions that did not end with a question mark to ensure clarity and consistency.
        - Converted answer strings into actual lists and extracted the correct answer using its index.
        - Formatted each sample into a structured prompt containing the context, question, and expected answer.
    - **Tokenization**
        - Used the tokenizer from `mistralai/Mistral-7B-Instruct-v0.1` to process the input prompts.
        - Applied padding using the end-of-sequence token and truncated to 256 tokens for memory control.
        - Set input IDs as labels to align with causal language modeling objectives.
        - Ensured tokenization followed the model's expected format for effective next-token prediction.
    - **Training**
        - Loaded Mistral model in 4-bit precision using BitsAndBytes for efficient GPU memory use.
        - Applied LoRA via PEFT, tuning only specific layers (query and value projections).
        - Trained using AdamW optimizer with a learning rate of 2e-5 and gradient accumulation.
        - Performed validation after each epoch and saved model checkpoints locally.

4. Results

4.1. Summarization Model

Both were tested on CNN-DailyMail News Text Summarization

-	Paper Results			Implemented Results		
-	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-1	ROUGE-2	ROUGE-3
F1	0.313	0.193	0.262	0.3378	0.1435	0.2429
Percision	0.388	0.275	0.289	0.2650	0.1115	0.1899
Recall	0.324	0.132	0.199	0.4931	0.2140	0.3565

4.2. Question-Answer Generation Model

-	Paper Results		Implemented Results
-	LSTM	BERT	Mistral
Cosine Similarity	0.22	0.55	0.3112
Rouge	0.12	-	-

4.3. Context Answer Generation Model

-	Paper Results		Implemented Results
-	LSTM	BERT	Mistral
Cosine Similarity	0.22	0.55	0.97