Intelligent Meeting Transcript Summarizer

—

By Vivek Singh

Problem Statement

- Modern professionals attend numerous long meetings with unstructured transcripts that are hard to revisit for insights.
- Manual summarization is time-consuming and often inconsistent, creating bottlenecks in information processing workflows
- Need for automated systems to generate general overviews and extract query-specific information from meeting transcripts efficiently

Project Motivation

1 Industry Trends

Remote work has significantly increased meeting volume, with information buried in transcripts that are difficult to process manually and efficiently retrieve

Solution Benefits

Transformer models like BART and T5 offer scalable solutions for query-specific summaries that support targeted knowledge retrieval from meetings

Project Introduction

Task Definition : Meeting summarization represents an advanced NLP task involving text generation using transformer architectures for high-quality summaries.

Technical Approach : Data preprocessing using QMSUM dataset with fine-tuning of four transformer models and comprehensive evaluation frameworks.

Implementation Scope : Complete deployment pipeline including lexical and semantic metrics evaluation via FastAPI and Docker containerization for production use.

Dataset Overview

QMSUM Dataset

Query-based Multi-domain Meeting Summarization dataset containing transcripts, summaries, queries and answers with metadata

Data Structure

Training set with 162 meetings, validation and test sets with 35 meetings each, supporting both general and query-focused summarization

Dataset Preparation Framework

Data Integration

Merged JSON files into standardized jsonl format and removed noise including unnecessary speaker tags for clean processing

Format Creation

Created normal format mapping full transcripts to summaries and gold format for query-specific span processing

Training Enablement

Structured data to enable both general meeting summarization and focused query-based summarization training workflows

Dataset Preparation Framework

Baseline Models

ART-base for general summarization tasks.

T5-base unified text-to-text

Standard sequence handling

Foundation architectures

Specialized Models

BART-large-XSum for extreme

BART-large-CNN multi-sentence

Domain-specific training.

Enhanced summarization

Selection Criteria

Pretraining data compatibility

Sequence length handling

Meeting domain suitability

Performance benchmarks

Dataset Preparation Methods

Model Name	Preprocessing Method	Input Handling	Length Limit	Notes
BART-base	Chunking	Segmented	512	Length constraints
T5-base	Chunking	Segmented	512	Token limits
BART-large-XSum	Truncation	Direct	1024	No chunking
BART-large-CNN	Full-length	Complete	Variable	Maximum context

Model-Specific Preprocessing Analysis

Each transformer model required customized preprocessing approaches based on architectural constraints and training objectives. BART-base and T5-base models required chunking due to input length limitations, processing transcripts in smaller segments to accommodate token limits. BART-large-XSum utilized truncation to 1024 tokens without chunking, suitable for concise summarization tasks. BART-large-CNN handled full-length transcripts without chunking, maximizing context retention for comprehensive meeting summarization and enabling better performance on longer, complex meeting discussions.

Results and Observations

Model	ROUGE-L	BERTScore	Key Observations	Performance Notes
BART-base	21	**84**	Short summaries	Context missed
T5-base	20	83	High variability	Inconsistent
BART-large-XSum	**29.1**	86.0	Fluent output	Hallucinations
BART-large-CNN	**29.7**	88.2	Balanced faithful	Best performance

Model Performance Analysis

Comprehensive evaluation revealed significant performance variations across transformer models. BART-base achieved ROUGE-L of 21 and BERTScore of 84 but produced short summaries with missed context. T5-base showed similar metrics with high variability and inconsistent outputs. BART-large-XSum demonstrated improved performance with ROUGE-L of 29.1 and BERTScore of 86.0, generating fluent summaries but exhibiting hallucination issues. BART-large-CNN emerged as the top performer with ROUGE-L of 29.7 and BERTScore of 88.2, providing balanced, faithful, and coherent summaries with optimal semantic and structural fidelity for meeting summarization tasks.

Final Model Selected

88.2%

BART-large-CNN Performance

Best semantic similarity score with balanced, faithful, coherent summaries and fewer hallucinations for real-world applications

Deployment Overview

Backend System

FastAPI Framework

RESTful API with comprehensive endpoint architecture design

Service Endpoints

Single, batch, and file upload summarization capabilities

Implementation Components

Frontend Interface

Jinja2-based minimal user interface design

Container System

Docker containerization for deployment

Local Deployment

Docker Desktop integration for development

Cloud Infrastructure

AWS ECS and App Runner support

Benefits

Scalability

Horizontal scaling with cloud-native architecture

Production Ready

Modular design with enterprise-grade deployment capabilities

3	2	100%
API Endpoints	Deployment Options	Containerized

API Endpoints

Single Summarization

- POST /summarize endpoint for individual transcripts
- JSON payload with meeting text
- Real-time processing and response
- Customizable summary length parameters

Batch Processing

- POST /batch-summarize for multiple meetings
- Array of transcript objects
- Parallel processing optimization
- Progress tracking capabilities

File Upload

- POST /upload for document processing
- Supports multiple file formats
- Automatic text extraction
- Download summarized results

◇ Project File Structure & Purpose

```
Intelligent_Meeting_Summarizer/
├── app/
│   ├── model/                          # Directory for saved/finetuned BAR
T models
│   ├── __pycache__/                    # Python cache files (ignored in lo
gic)
│   ├── bart_large.py                   # Summarization logic using BART La
rge XSum
│   ├── bart_large_cnn.py               # Summarization logic using BART La
rge CNN
│   ├── main.py                         # FastAPI routes, endpoints, and ap
p logic
│   ├── schemas.py                      # Pydantic models for request/respo
nse
├── static/                             # Static assets (CSS, JS)
├── templates/                          # HTML templates (for `/ui` endpoin
t)
│   └── index.html
├── output/                             # Optional folder for saving summar
ies
├── requirements.txt                    # Python dependencies
├── Dockerfile                          # Docker configuration
```

◇ FastAPI Endpoints

Below are the endpoints implemented in main.py and their purposes:

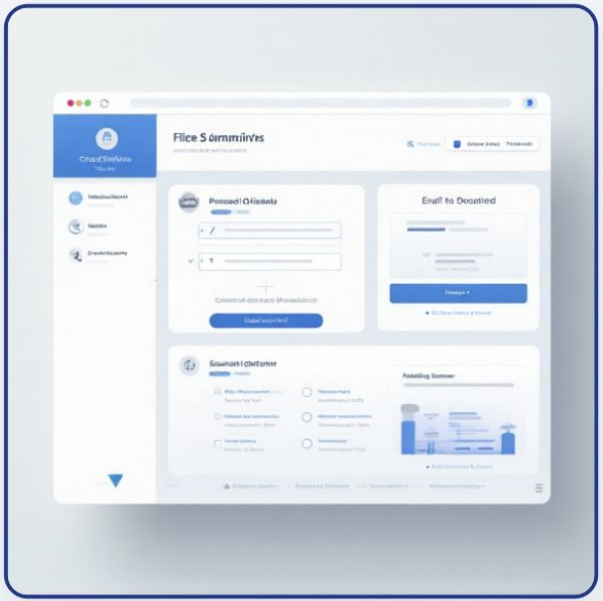
- GET / — Root endpoint. Returns a welcome message.
- GET /ui — Serves the index.html UI for uploading transcripts and viewing summaries.

BART Large (XSum):

- POST /bart_large — Summarizes a single transcript using the bart_large model.
- POST /bart_large/batch — Summarizes a batch of transcripts.
- POST /bart_large/batch_upload_json — Upload a .json file containing batch transcripts and get summaries.

BART Large CNN:

- POST /bart_large_cnn — Summarizes a single transcript using the bart_large_cnn model.
- POST /bart_large_cnn/batch — Summarizes a batch of transcripts.
- POST /bart_large_cnn/batch_upload_json — Upload a .json file with transcript data and get batch summaries.



Frontend Implementation

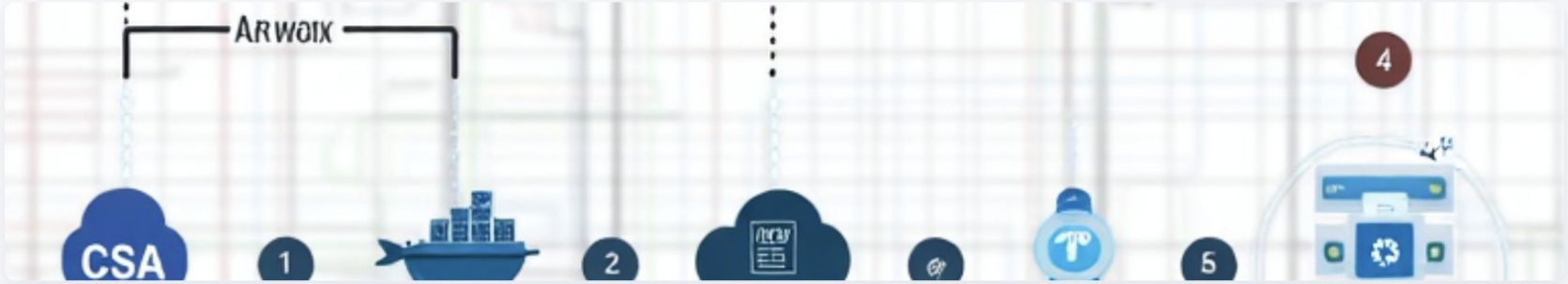
Jinja2 Templates

Server-side rendering with dynamic HTML generation and responsive design for cross-platform compatibility and user experience

User Interface

Minimal, clean interface design with file upload functionality, real-time processing status, and downloadable summary results

Docker Containerization Framework



1 — 2 — 3

Container Setup

Multi-stage Dockerfile with Python base image, dependency installation, and optimized layer caching for efficient builds

Environment Configuration

Environment variables for model paths, API keys, and configuration settings with secrets management for production deployment

Deployment Strategy

Local Docker Desktop development environment and cloud deployment via AWS ECS with auto-scaling and load balancing

◇ Deploying to AWS :

After successfully pushing the Docker image to Docker Hub, the next step is to deploy it AWS. This can be done using **Amazon ECS (Elastic Container Service)** with **Fargate** for serverless and scalable deployment environment. The deployment workflow involves creating a new ECS **Cluster**, followed by defining a **Task Definition** that references the Docker image from Docker Hub. You configure CPU, memory, and networking settings in the task. Then, an ECS **Service** is created to run the task continuously, and it can be attached to an **Application Load Balancer** to make the API publicly accessible.

Alternatively, **AWS App Runner** can be used for a simpler experience. You connect your Docker Hub repository to App Runner, specify the container port (8000), and deploy directly. App Runner automatically manages scaling, HTTPS, and health checks, making ideal for containerized FastAPI applications.

◇ Conclusion:

In this project, we successfully developed an intelligent, end-to-end meeting summarization system that combines the power of transformer-based models with scalable deployment infrastructure. By systematically preprocessing the QMSUM dataset, fine-tuning multiple models (BART-base, T5-base, BART-large-XSum, BART-large-CNN), and deploying the best-performing model via a FastAPI-Docker pipeline, we demonstrated a practical solution for condensing lengthy and unstructured meeting transcripts into concise, semantically faithful summaries. The BART-large-CNN model emerged as the most reliable choice, offering a balance between coherence, informativeness, and minimal hallucinations. This work sets a solid foundation for future improvements, including real-time summarization, multi-lingual support, and tighter integration with productivity tools for seamless knowledge management in modern workplaces.

Thank You

Questions and Discussion

We appreciate your attention and welcome any questions about the Intelligent Meeting Summarizer project