Personalized News Summarizer: An AI-Powered System for Automated Article Extraction, Summarization, and Evaluation

Github https://github.com/Arph003/personalized-news-summarizer.git

Team Name: GoodTeam

Team Members: Zhiqi Li & Yuan Tian

Abstract:

This project presents Personalized News Summarizer, an AI-powered system designed to streamline information consumption by automatically extracting, summarizing, and evaluating online news articles. Modern readers face significant information overload, and manually processing lengthy articles is inefficient and often impractical. To address this challenge, our system integrates web scraping, transformer-based abstractive summarization, and quantitative evaluation into a unified, user-friendly web application.

The system retrieves article content directly from user-provided URLs using a custom scraping module built with requests and BeautifulSoup. Summaries are generated through a pre-trained T5-small transformer model, selected for its balance between performance and computational efficiency. To assess summary quality, the system computes ROUGE-1, ROUGE-2, ROUGE-L, and BERTScore metrics, supplemented by additional measures such as vocabulary coverage, word counts, and runtime. These results are presented through a clean web interface built with Flask and modern frontend components.

Our findings demonstrate that the system effectively condenses long-form articles into concise summaries while preserving key semantic content. The evaluation metrics reveal meaningful insights into model behavior, highlighting strengths and limitations of abstractive summarization. Overall, this project illustrates the practical integration of generative AI techniques to enhance everyday news consumption and provides a framework for future extensions involving personalization, model comparison, and improved factuality validation.

Introduction:

In today's digital environment, individuals are exposed to an overwhelming volume of online information. News outlets, blogs, and social media platforms continuously produce content, making it increasingly difficult for readers to stay informed without dedicating significant time and cognitive effort. As a result, tools that can automatically distill long articles into concise, accurate summaries have become essential for efficient information consumption. However, many existing summarization solutions operate as "black boxes," lack

transparency, or fail to provide meaningful evaluation metrics to help users understand the quality of the generated summaries.

This project addresses these challenges by developing *Personalized News Summarizer*, an AI-powered web application that retrieves news articles from user-provided URLs, generates abstractive summaries, and evaluates their quality through established NLP metrics. The objective is to create a system that not only summarizes content effectively but also provides users with deeper insight into the summarization process through quantitative analytics such as ROUGE and BERTScore.

Our approach integrates three core components: a robust article extraction pipeline, a transformer-based summarization model, and a clean, interactive user interface. By combining modern natural language processing methods with practical web development, the project demonstrates how generative AI can support personalized knowledge management. The system is lightweight, extensible, and designed to serve as a foundation for future enhancements, including user preference modeling, topic filtering, and multi-document summarization. Through this work, we aim to illustrate the value of AI-driven summarization in making daily information consumption more efficient, transparent, and user-centric.

Data:

Unlike traditional machine learning projects that rely on static, pre-defined datasets, this project operates on dynamic, user-specified web data. The system extracts textual content directly from online news articles using URLs provided by the user. This design ensures flexibility, allowing the application to handle a wide range of article formats, domains, and writing styles.

Data Source

The primary data source is the publicly accessible HTML content of online news websites. Articles are retrieved through HTTP requests and parsed using the BeautifulSoup library. Although users may input any valid URL, our testing primarily utilized news articles from reputable outlets such as CNN, BBC, and IBM's technology blog. These articles typically range from 500 to 5,000 words, providing ample variation for evaluating model performance.

Data Characteristics

The extracted data contains:

- Article title

- Main body text (paragraph-level content)
- Occasionally, metadata such as author or publication date (when available)

Since web pages often include unrelated elements such as navigation bars, advertisements, or scripts, the system applies heuristic-based cleaning to isolate the core text. This ensures that only meaningful content is fed into the summarization model.

Preprocessing Steps

Prior to summarization, the system performs:

1. HTML parsing to extract textual paragraphs
2. Whitespace normalization to ensure clean model input
3. Length filtering to remove extremely short or empty text fragments
4. Concatenation of article paragraphs into a unified document

Data Flow Diagram

A simplified overview of the data pipeline is shown below:

User Input URL

↓

HTTP Request → Raw HTML

↓

BeautifulSoup Parsing

↓

Content Extraction & Cleaning

↓

Clean Text Document

↓

Summarization Model (T5-small)

↓

Evaluation Metrics (ROUGE, BERTScore)

This dynamic data pipeline enables the system to generalize across diverse writing styles and article structures, demonstrating the real-world applicability of the summarization and evaluation modules.

Experimental Setup

This section describes the technical configuration, system architecture, and evaluation methodology used to develop and test the *Personalized News Summarizer*. The goal is to provide sufficient detail for reproducibility and clarity regarding the system's design choices.

1. System Architecture

The project is implemented as a modular, end-to-end pipeline consisting of three primary components:

1. **Web Scraper**
    - Retrieves article content from user-provided URLs.
    - Built using `requests` for HTTP communication and `BeautifulSoup` for HTML parsing.
    - Includes heuristic-based cleaning to isolate core article text.
2. **Summarization Engine**
    - Utilizes the pre-trained **T5-small** transformer model from Hugging Face.
    - Performs abstractive summarization by converting the cleaned article text into a concise natural-language summary.
    - Runs inference using PyTorch with hardware acceleration (Apple MPS when available).
3. **Evaluation Module**
    - Computes ROUGE-1, ROUGE-2, ROUGE-L, and **BERTScore** to assess summary quality.
    - Reports additional metrics: summary length, compression ratio, vocabulary coverage, and runtime.
    - Output is structured and returned through JSON for front-end rendering.

These components are orchestrated through a Flask API, while a lightweight HTML/JavaScript front-end provides real-time interaction.

2. Software and Libraries

The system was developed using the following software stack:

- **Python 3.12**
- **Flask 3.1** (web server and API)
- **Transformers 4.57** (Hugging Face model pipeline)
- **PyTorch 2.9** (model execution, MPS-enabled)
- **BeautifulSoup4** (HTML parsing)
- **Requests** (web content retrieval)
- **ROUGE-score** (token-based evaluation)
- **BERTScore** (semantic similarity evaluation)

These libraries were installed within an isolated Python virtual environment to ensure dependency consistency.

3. Model Configuration

The summarization component uses:

- Model: t5-small
- Generation settings:
    - max_length = 130
    - max_new_tokens = 256
    - no_repeat_ngram_size = 2
    - temperature = 1.0
    - top_k = 50
    - top_p = 1.0

These parameters were chosen to balance summary conciseness, fluency, and inference speed.

4. Evaluation Metrics

To quantitatively assess summarization quality, the system computes:

ROUGE Metrics

- **ROUGE-1**: Unigram overlap
- **ROUGE-2**: Bigram overlap

- **ROUGE-L**: Longest common subsequence
  Used to approximate lexical similarity between the generated summary and the reference document.

BERTScore

- Uses contextual embeddings to measure semantic similarity.
- Reports **Precision**, **Recall**, and **F1-score**.
- Useful for abstractive summarization because it evaluates meaning rather than surface overlap.

Additional Metrics

- **Original vs. summary word count**
- **Compression ratio**
- **Vocabulary coverage**
- **Runtime (ms)** for performance benchmarking

These metrics allow a more holistic analysis of model behavior.

5. Experimental Procedure

1. Select a set of representative online news articles.
2. Input each URL into the application.
3. The scraper retrieves and processes article text.
4. The summarizer generates an abstractive summary.
5. Evaluation metrics are computed automatically.
6. Results are visualized in the web interface and stored for analysis.

This standardized procedure ensures consistency across all experiment runs.

Results

This section presents the quantitative performance of the *Personalized News Summarizer* across representative news articles. The goal is to evaluate how effectively the T5-small model condenses long-form online content while preserving semantic meaning. Results are reported using ROUGE, BERTScore, and additional system-level metrics such as compression ratio and runtime.

1. Quantitative Evaluation

Evaluation was performed on multiple news articles scraped from public sources. A representative example is shown below, using the IBM article *"What Are Foundation Models?"* as the test case.

Summary Quality Metrics

Metric	Score
ROUGE-1 (F1)	0.0437
ROUGE-2 (F1)	0.0414
ROUGE-L (F1)	0.0437
BERTScore (F1)	0.4062
Precision	0.7662
Recall	0.1993

Interpretation:

- ROUGE scores are low, which is expected for abstractive summarization because the generated summary compresses the article dramatically and uses new phrasing.
- BERTScore F1 of **0.40** indicates moderate semantic alignment with the original text, demonstrating that key ideas are preserved despite aggressive compression.
- High precision but low recall reflects that the summary captures essential concepts but omits many details from the long article.

2. Compression and Structural Metrics

Metric	Value

Original length (characters)	11,364
Summary length (characters)	265
Compression ratio	0.023
Original word count	1,643
Summary word count	41
Vocabulary coverage	0.9737
Runtime	4,426 ms

Interpretation

- A **97% reduction** in text size confirms the model's strong ability to condense long documents.
- Vocabulary coverage remains high, showing stable tokenization behavior across diverse content.
- Runtime under 5 seconds demonstrates that the system is responsive even without GPU acceleration, making it practical for real-time use.

3. Example Summary Output

Below is the actual summary generated by the system for the IBM article:

> **"AI Models are artificial intelligence models trained on vast, immense datasets. They serve as the base or building blocks for crafting more specialized applications. Their flexibility and massive size set them apart from traditional machine learning models."**

This summary successfully captures the core message of the original article: the definition, purpose, and distinguishing characteristics of foundation models.

4. Cross-Article Consistency

Across all tested articles, the system demonstrated consistent behavior:

- Summaries remained **short, fluent, and coherent**.
- ROUGE scores remained low (a known limitation for abstractive summarizers).
- BERTScore remained moderate across articles, showing stable semantic retention.
- Runtime varied only slightly depending on article length.

These results confirm that the pipeline generalizes well across different news domains.

Discussion

The results demonstrate that the Personalized News Summarizer successfully integrates article scraping, abstractive summarization, and automated evaluation into a cohesive end-to-end system. The generated summaries effectively capture the central themes of long-form news articles, significantly reducing reading time while preserving core meaning. The evaluation metrics provide additional insight into how the summarization model behaves across real-world content.

Although ROUGE scores appear low, this behavior is expected for transformer-based abstractive summarization models such as T5-small. ROUGE measures surface-level lexical overlap, which penalizes summaries that rephrase content or use different wording—a core characteristic of abstractive methods. In contrast, the BERTScore results show moderate semantic alignment between summaries and the original articles, indicating that the model captures key ideas even when phrasing differs substantially. The high precision but lower recall suggests that the summaries tend to include only the most essential concepts while omitting many secondary details.

Several challenges emerged during experimentation. Web articles vary widely in structure, and scraping requires robust handling of inconsistent HTML layouts and irrelevant textual elements. While the scraper performs well on most pages, highly structured or script-heavy websites may still require additional tuning. Another limitation concerns summary brevity: the model tends to produce extremely short summaries, which improves readability but sometimes sacrifices nuance. Adjusting generation parameters or employing larger models could improve coverage and contextual richness.

Nevertheless, the system demonstrates the practicality of combining generative AI with real-time web interaction. By providing interpretable evaluation metrics alongside summaries, the application enhances transparency—an important consideration when deploying AI systems for information consumption. The project highlights the potential of lightweight transformer

models to support personalized news reading, while also illustrating the importance of evaluation frameworks that go beyond surface-level metrics.

Potential Future Directions

There are several meaningful directions to extend this project beyond its current implementation. First, the summarization module could be upgraded with more advanced transformer models—such as PEGASUS-large or T5-base—or fine-tuned on news-specific datasets to improve coherence and factual accuracy. Adding factual consistency checks would further reduce hallucinations and strengthen summary reliability.

The application could also be enhanced to support multi-article aggregation, allowing users to input multiple URLs and receive a unified "topic digest" summarizing common themes and differences across sources. This feature would significantly improve its value for users who follow complex or rapidly evolving news events.

Personalization represents another promising direction. By storing user preferences—such as favored topics, summary length, or writing style—the system could generate customized summaries tailored to individual reading habits.

Finally, several system-level improvements could move the tool closer to production readiness, including a more polished frontend, database-backed user profiles, Docker containerization, or deployment on cloud platforms. Each of these extensions would strengthen the overall usability, scalability, and real-world impact of the Personalized News Summarizer.

Conclusion

This project presents a complete workflow for building a personalized, AI-powered news summarization system that helps users process large volumes of information more efficiently. By combining web scraping, transformer-based abstractive summarization, and automated evaluation metrics, the system provides real-time summaries that are concise, readable, and grounded in modern NLP techniques. The integration of ROUGE and BERTScore further strengthens the system by offering quantitative insight into summary quality and model behavior.

Our experiments demonstrate that even lightweight pre-trained models, such as T5-small, can generate coherent summaries with minimal computational overhead, making the solution practical for everyday use. The modular architecture—built with Flask, BeautifulSoup, and Hugging Face Transformers—ensures that each component is transparent and easily extensible.

While the system performs well as a functional prototype, limitations remain, including overly short summaries and limited factual consistency checks. Nonetheless, the project establishes a solid foundation for future enhancements such as multi-article summarization, personalization features, improved factual verification, and deployment on cloud platforms.

Overall, the Personalized News Summarizer successfully showcases how generative AI can streamline information consumption and support productivity in a world saturated with digital content.