News Summarizer using Retrieval Augmented Generation

Manya Goel, 2310110173 Dhanya Girdhar, 2310110463 Raashi Sharma, 2310110566

***Abstract*—The rapid growth of digital news has made it overwhelming for users to find relevant news out of the hundreds repetitive, lengthy and difficult-to-filter articles. Identifying the most relevant news has become a major challenge. This project presents a module IR- and NLP-based system that retrieves the most relevant news article, filters them through gap-based relevance ranking. It generates concise, factual and query focused summaries. The system combines corpus building from news sources, retrieval-augmented summarization and follow-up answering to give accurate and factual answers to users.**

I. Introduction

With the widespread availability of the World Wide Web, the amount of information on the internet is growing at an exponential rate. While this provides users with unprecedented access to knowledge, it also makes it increasingly difficult to identify and consume relevant content, especially given today's fast-paced lifestyles. This has amplified the need for techniques that can abstract or condense information without losing essential meaning. News summarization presents the user with a shorter version of the data with only vital information. This helps them to understand the information faster.

Summarization Definition Natural Language Processing community has been investigating the domain of summarization for nearly the last half-century. Radev et al (2002) define summary as "text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that." Three main aspects of research on automatic summarization are delineated by this definition:

- Summaries may be produced from multiple websites or reports
- Summaries should preserve important information
- Summaries should be short

Manual summarization is a time-consuming process. The most important task in extractive text summarization is choosing the main sentences that would appear in the summary. Identifying such sentences is a truly challenging task. This problem is addressed by automatic summarization, which cleverly recognizes and extracts the most crucial sentences from a document. As a result, it is now more relevant in a variety of fields, such as content recommendation systems, digital news platforms, search engines, and document analysis tools.

The project focuses on IR-based summarization with user-centric features like query search, follow-up questions, top headlines, and interactive news-literacy games.

II. Literature Review

Document ranking is based on conventional models like TF-IDF and BM25. Robertson & Walker (1994) used probabilistic weighting to show how well BM25 captures term frequency and document rarity. Nevertheless, phrase-level semantics is not taken into account by these models, which only use unigram matching. By adding bigram-level scoring for headlines, our system expands on this.

A number of studies also make use of external knowledge and synonym dictionaries. In order to decrease vocabulary mismatch and increase retrieval accuracy, our project incorporates a lightweight synonym mapping (e.g., AI $\rightarrow$ artificial intelligence, US $\rightarrow$ United States).

While previous research focuses on specific issues-phrase extraction, document ranking, summarization, and quality assurance-our project is the first to combine all of these elements into a single IR pipeline. To create an end-to-end, traceable, and hallucination-free IR system, we suggest a hybrid unigram-bigram retrieval model, a grounded summarizer, and a document-cached follow-up QA module.

III. Objective

In today's fast-paced world, users don't have time to read everything. Many articles from different news sources are often lengthy, repetitive, or difficult to read. Our objective was to make news easily readable and manageable so that users, especially kids and teenagers, could fit it into their daily schedules. In addition to being informed engagingly and fascinatingly, we wanted people to be able to easily consume news in five minutes.

This will encourage consistent news-reading habits. Summaries would be produced from multiple websites or reports. People can cross-question on summaries to delve more in depth. It has been observed that gamification of a monotonous and boring task attracts users and makes them more consistent.

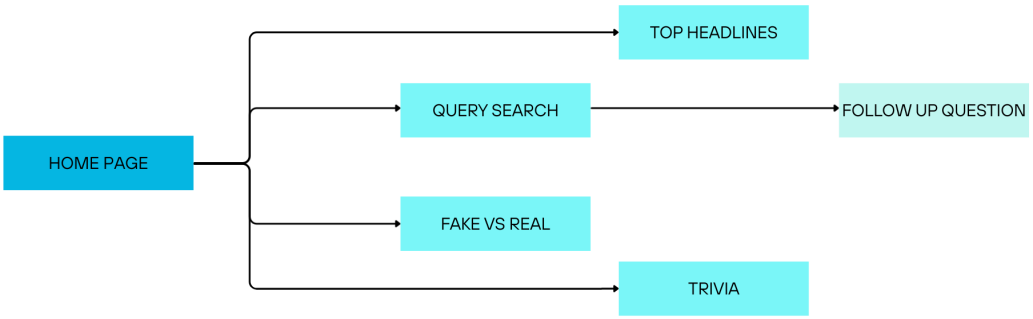

Fig. 1. Flow Chart

IV. PROPOSED MODEL

The proposed model contains interconnected modules that work to retrieve news, summarize information, and provide interactive features to users. Figure 1 depicts the workflow of our system. The following describes the architecture and functionality of each module:

1) Dataset Building Module:
   - Extracts the news from credible Indian sources
   - Uses GDELT API with Trafilatura-based webscraping
   - Creates the corpus for a time period

2) Headlines Retrieval Module:
   - Extracts top headlines using an API
   - Gets displayed on the home page
   - Each headline is presented along with its URLs and summaries.

3) Query-Based Search Module:
   - Accepts users' query
   - Processes the corpus through IR-based models
   - Retrieves the top matching news articles
   - Sends it to the summarization module

4) Summarization Module:
   - It takes retrieved documents as input
   - Converts long articles into a summary

5) Follow-Up Question Module:
   - Answers users' secondary queries
   - Searches only within the previously matched documents

6) Trivia Game Module:
   - Filters the corpus according to categories
   - Processes an entire corpus
   - Finds the top 3 most important sentences for each article
   - Uses Gemini API used to generate MCQs and fill in banks

7) Fake vs Real News Module:
   - Extracts only the headlines from the corpus
   - Creates a fake version of each headline
   - Users are presented with three options and asked to identify the real one
   - Tests users media literacy

V. METHODOLGY

Our system aims to retrieve headlines, generate summaries based on a user's query, handle follow-up questions in a conversation, and support interactive search-based applications. The overall pipeline is composed of three major layers:

- Document Preprocessing and Feature Construction
- Search and Ranking Engine (Hybrid Unigram–Bigram IR Model)
- Summarization and Follow-Up Question Answering Module

DOOCUMENT PROCESSING PIPELINE:

1) **Text Cleaning** This step removes encoding artifacts, "Published/Updated" metadata lines, location–date headers, URLs, punctuation, timezone markers, non-word symbols, percentage markers, and expands abbreviations (e.g., `U.S. → united states`).

2) **Tokenization and Lemmatization (spaCy)** Each cleaned document is processed using spaCy. Tokens are lowercased, stopwords removed, and words lemmatized for semantic consistency.

3) **Bigram Extraction (Title-Level)** Headlines are often phrase-driven, so adjacent bigrams are generated from the title field to support accurate query matching.

4) **LNC-Vector Construction (Content-Level)** For each document, clean content tokens are converted into LNC-weighted vectors.
   - Raw LNC weight:

$$\mathrm{LNC}(t) = 1 + \log(\mathrm{tf}_t)$$

- Vector normalization:

$$w_t = \frac{\text{LNC}(t)}{\sqrt{\sum_i \text{LNC}(i)^2}}$$

Each document stores:

- `vector` - normalized unigram vector
- `term_lnc` - raw LNC weights
- `title_bigrams` - raw bigram tokens
- `title_bigram_weights` - weighted bigram vector

This LNC representation forms the foundation of the hybrid unigram-bigram search engine.

SEARCH AND RANKING PIPELINE:

1) **Query Processing and Synonym Expansion**

The User queries are cleaned and tokenized using the same preprocessing as used in documents. In addition, a synonym-expansion map is used to reduce mismatch:

- us,usa $\rightarrow unitedstates$
- pm $\rightarrow primeminister$
- ai $\rightarrow artificialintelligence$
- etc

2) **Query Vector Construction (Unigrams + Bigrams)**

Two Query vectors are computed:

- Unigram Ltn vector
- Bigram vector

Bigrams/Terms not found in corpus vocabulary are ignored for precision

3) **Hybrid Relevance Scoring**

Each document is scored using a weighted combination:

$$\text{Relevance} = \alpha \cdot \text{content\_score} + (1 - \alpha) \cdot \text{bigram\_score}$$

Where:

- `content_score`: cosine similarity between query unigrams and document LNC vector
- `bigram_score`: similarity between query bigrams and document title-bigram vector

Default values:

- `content_weight`: 0.65
- `bigram_weight`: 0.35

This ensures phrase recognition while doing content-matching.

4) **Dynamic Pruning**

This method filters out irrelevant documents based on the rule:

$$\text{overlap\_weight} \geq \alpha \cdot \text{max\_query\_weight}$$

In addition to that, a gap-based cutoff removes documents when it detects large drops in relevance, improving precision

5) **Output**

The system returns:

- title
- URL
- relevance score sub-scores

This feeds into the summarization and follow-up modules.

SUMMARIZATION AND FOLLOW-UP QUESTIONS:

1) **Query-Based Summarization (LLM, RAG-Based)** This component uses Retrieval-Augmented Generation (RAG) workflow to produce query-specific summaries.

a) **Top-K Retrieval** The System retrieves the most relevant documents for user query.
b) **Evidence Construction** A compact evidence block is created using document titles and trimmed content.
c) **Grounded LLM Prompting** A hallucination-free prompt is sent to Gemini (gemini-2.5-flash) requiring that only retrieved document text may be used and no external information may be introduced. The LLM generates a concise summary grounded entirely in the retrieved evidence which ensures summaries are factually supported.

2) **Follow-Up Question Handling**

a) **TF-IDF Passage Extraction:** Each document is split into sentences, followed by a TF-IDF vectorizer, which is fitted on all sentences and follow-up query. Top-Scoring sentences (using Cosine-Similarity) above a threshold are selected as relevant.
b) **IR Answering:** The extracted passages are used and system sends a controlled prompt to give a short, factual answer.

VI. EXPERIMENTS AND RESULTS

Below are the screenshots of the system's output at different stages - the initial search results, the generated summary, and the follow-up responses.

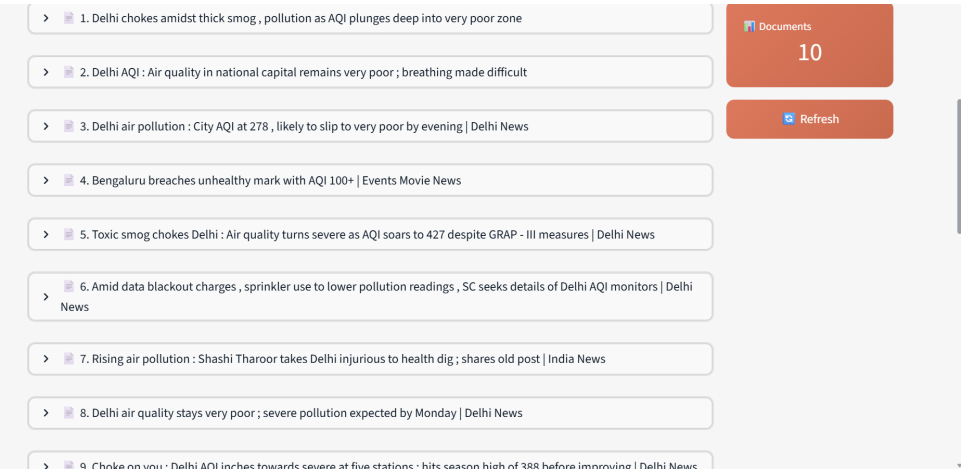

Fig. 2. Relevant documents retrieved for the input query.

The first screenshot, shown in Fig. 2, displays the search interface and the results returned for the query.

Next, we present the summary generated by the summarization module. As seen in Fig. 3, the summarizer produces an extractive summary derived solely from retrieved document context.

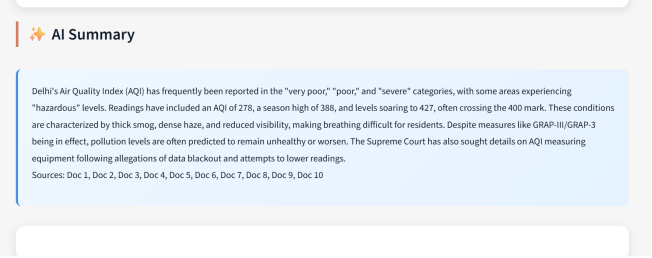

Fig. 3. Generated summary using the extractive summarization module.

Then, we have the follow-up question workflow, shown in Fig. 4.

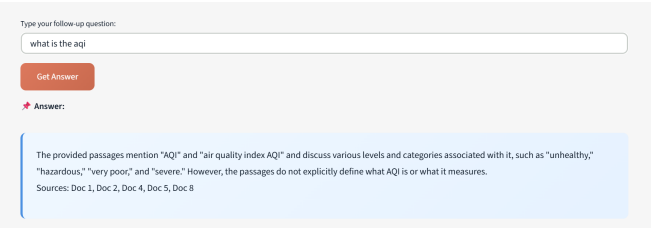

Fig. 4. Follow-up query handling and generated answer.

VII. CONCLUSIONS AND LIMITATIONS

We were able to automatically summarize news articles and compare the generated summaries to determine which scoring parameters produced better results. During this process, we refined our initial approach by leveraging the fact that our corpus consisted exclusively of news articles. Initially, we applied TF–IDF scoring to both headlines and article bodies. However, after reviewing preliminary results, we noticed that users' queries are often phrase-like and require matching terms to appear close to each other. To account for this, we incorporated bigram-based scoring to better capture query relevance. Since applying bigrams across the entire corpus was computationally expensive, we restricted bigram matching to the headline, which significantly improved relevance without affecting performance.

VIII. LIMITATIONS

1) Currently, we can only retrieve news for one month due to the limited API access restriction
2) Free versions of LLM APIs, such as Groq and Gemini, allow a very low number of requests per minute, which resulted in slower processing.
3) The corpus can only be generated by running a code and is highly local and limited. There is no real-time scraping pipeline and regular corpus updation.

REFERENCES

[1] Hang Yu, Jiawei Han, "Survey of Query-Based Text Summarization"

[2] Entesar Alanzi, Safa Alballaa, "Query-Focused Multi-document Summarization Survey".

[3] Prakhar Sethi1, Sameer Sonawane2, Saumitra Khanwalker3, R. B. Keskar4, "Automatic Text Summarization of News Articles".

[4] Ojas Ahuja1, Jiacheng Xu1, Akshay Gupta1, Kevin Horecka2, Greg Durrett1, "ASPECTNEWS: Aspect-Oriented Summarization of News Documents".

[5] Sharma, G., Sharma, "Automatic Text Summarization Methods".

[6] Kung-Hsiang Huang1, Philippe Laban2, Alexander R. Fabbri2, "Embrace Divergence for Richer Insights: A Multi-document Summarization Benchmark and a Case Study on Summarizing Diverse Information from News Articles".