An Interpretable Retrieval-Augmented Generation (RAG) System with Local Knowledge Graph Explanations

Technical Report

Team FPS

Pratyaksha Jha Somita Agarwal Freya Sheth

B.Tech, Indian Institute of Technology Guwahati

January 14, 2026

Abstract

Traditional Retrieval-Augmented Generation (RAG) systems focus on improving answer accuracy by grounding language models in retrieved documents, but they often fail to provide transparency regarding why a particular answer was generated. This limits trust, verifiability, and usability in high-stakes or academic settings. By utilizing a Local Contextual Entity Graph, the system provides a transparent pipeline that not only answers queries but also visualizes the metrics, entities, and evidence used to derive that answer. The system is designed to be client-side and privacy-preserving making sure that the data never leaves the user's machine. The system prioritizes clarity, traceability, and grounding over complex reasoning or large-scale optimization.

Contents

System Overview

The pipeline follows a modular design centered on client-side processing to ensure data privacy and immediate interpretability.

1. **Document Ingestion:** Supports PDF, DOCX, and JSON using `pdf.js` and `mammoth.js`. Documents are fragmented into overlapping chunks to preserve local context. A **Heuristic Denoising** layer filters out diagram metadata, page numbers, and low-alpha ratio strings to ensure the generation context remains human-readable.
2. **Retrieval Engine:** A term-frequency scoring algorithm with a specific **Factual Boost** for numerical data.
3. **Local Entity Extraction:** Pattern-based NLP to identify metrics and named entities within the retrieved subset.
4. **Extractive Synthesis:** A grounding-first approach that generates answers by selecting and ranking the most relevant source sentences.
5. **Explanation Layer:** A structured output consisting of color-coded entity tags and interactive evidence blocks.

Each component is intentionally designed to be modular, interpretable, and easy to reason about.

Query Processing and Retrieval

Query Processing and Retrieval

The system employs a Term-Frequency weighted retrieval method to identify the most relevant context for a given user query. The relevance score S for a document D given a query Q is calculated using the following objective function:

$$S(Q, D) = \frac{\sum_{t \in Q}(\text{count}(t, D)) + \text{Coverage}(Q, D) + \text{PhraseBoost}}{\ln(|D| + 1)} + 1_{\text{fact}}(D) \tag{1}$$

Where:

- **Coverage:** A boost of $+2$ for every unique query term found in the document.
- **PhraseBoost:** A static boost of $+5$ if the exact query string exists as a contiguous phrase.
- **Normalization:** The score is scaled by the natural log of the document length to prevent long documents from dominating results.

Design Choice: Keyword-based search was selected over dense vector embeddings for this prototype to ensure **exact-match transparency**. In explainable systems, it is critical that users can audit the retrieval process by identifying the specific lexemes that triggered a document's selection, thereby eliminating the "black box" nature of latent semantic spaces.

Entity Extraction and Graph Construction

Entity Extraction

Entities are extracted using high-performance Regular Expressions (Regex) to maintain system speed and transparency.

- **Technique:** Pattern-based Extraction.
- **Categories:**
  - **METRIC:** Identified via patterns of numbers followed by units like %, °C, million, or billion.
  - **ENTITY:** Identified by Title Case patterns (e.g., "Paris Agreement") for sequences of 1-3 capitalized words.

  Graph Logic:
  - **Nodes:** The extracted Entities and Metrics.
  - **Edges:** Implicitly defined by co-occurrence within the same retrieved text chunk. This creates a "Local Context Graph" that represents the specific world-view of the retrieved evidence, avoiding "hallucinated" connections from global pre-training.

Relationship Construction

A local knowledge graph is constructed by mapping relationships between the query, extracted entities, and specific sentences:

- **Sentence-Level Relevance:** Sentences are scored based on the density of query terms and extracted entities.
- **Evidence Mapping:** The top 3 sentences per retrieved document are linked to the answer as "Eyes-on" evidence.

Graph Representation

The graph is presented as a Relational Map where entities serve as nodes, and their co-occurrence within high-relevance 'Evidence Blocks' constitutes the edges of the local context:

- **Nodes:** `{text, type: METRIC | ENTITY}`
- **Edges:** Implicit links between entities and their source sentences, presented as highlighted evidence blocks.

Answer Generation

Context Construction

The answer generation context consists of retrieved text chunks and highlighted sentences containing key entities.

Generation Strategy

To meet the requirement of strict interpretability, the system utilizes **Extractive Synthesis** rather than abstractive generation. The system:

1. Scans retrieved chunks for sentences containing query keywords and extracted entities.
2. Ranks these sentences based on a local relevance score.
3. Concatenates the top 4 sentences to form a factual response.

The "I Don't Know" State

A key feature of our design is the "No Answer Found" state. If no direct overlap exists between the query and the knowledge base, the system refuses to generate a response. This is a critical safety mechanism for transparent RAG, preventing the fabrications common in standard language models.

Preliminary Results & Experiments

The system outputs a structured explanation alongside the final answer.

Test Case: Query from an uploaded book

- **Query:** "spfa algorithm"
- **Retrieval:** A negative cycle can be detected using the Bellman–Ford algorithm by running the algorithm for n rounds. Note that this algorithm can be used to search for a negative cycle in the whole graph regardless of the starting node. 125 SPFA algorithm The SPFA algorithm (”Shortest Path Faster Algorithm”) [20] is a variant of the Bellman–Ford algorithm, that is often more efficient than the original algorithm. The SPFA algorithm does not go through all the edges on each round, but instead, it chooses the edges to be examined in a more intelligent way. (score : 5.9)
- **Evidence from Retrieved Documents:** Sentences and files from which entities and relations were derived.

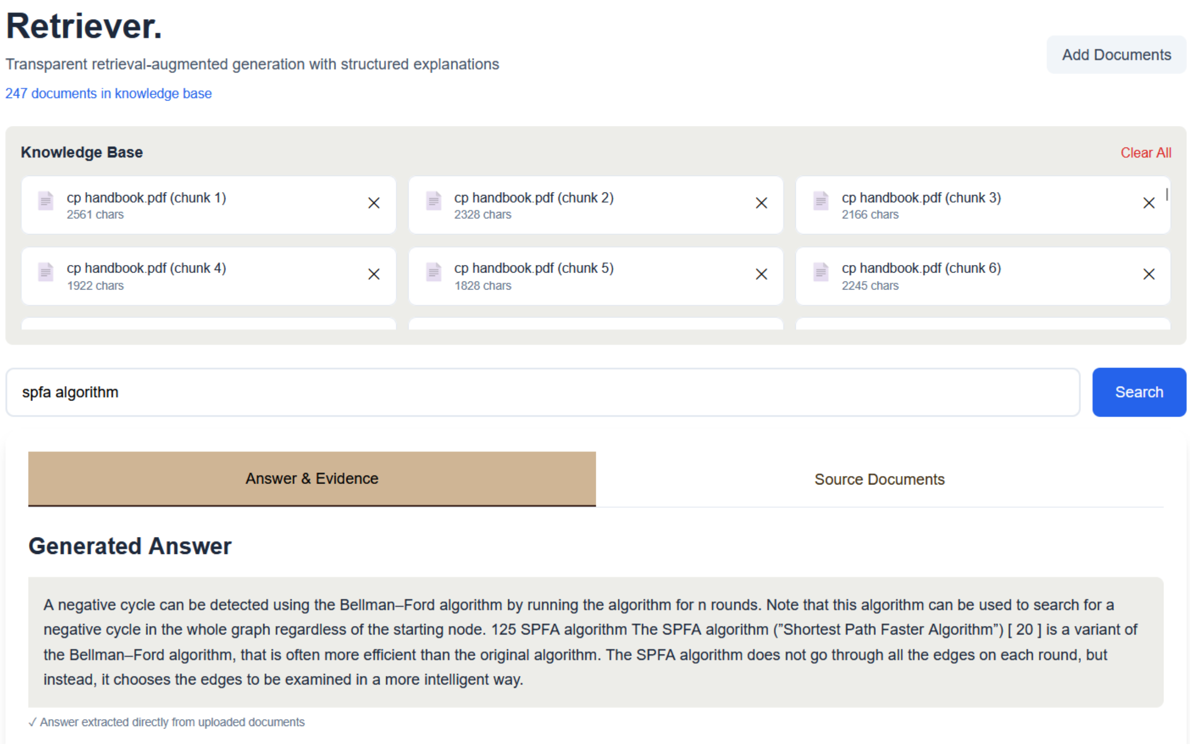

Figure 1: Sample input and output.

Interpretability and Transparency

Interpretability is achieved through:

- **Evidence Highlighting:** The system displays the exact sentences from the source documents that contributed to the score.
- **Structured Entity Tags:** Entities are color-coded (Blue for Metrics, Green for Entities) so users can see the "building blocks" of the information at a glance.
- **Relevance Scoring:** Each source document is displayed with its calculated score, allowing users to evaluate why one document was preferred over another.

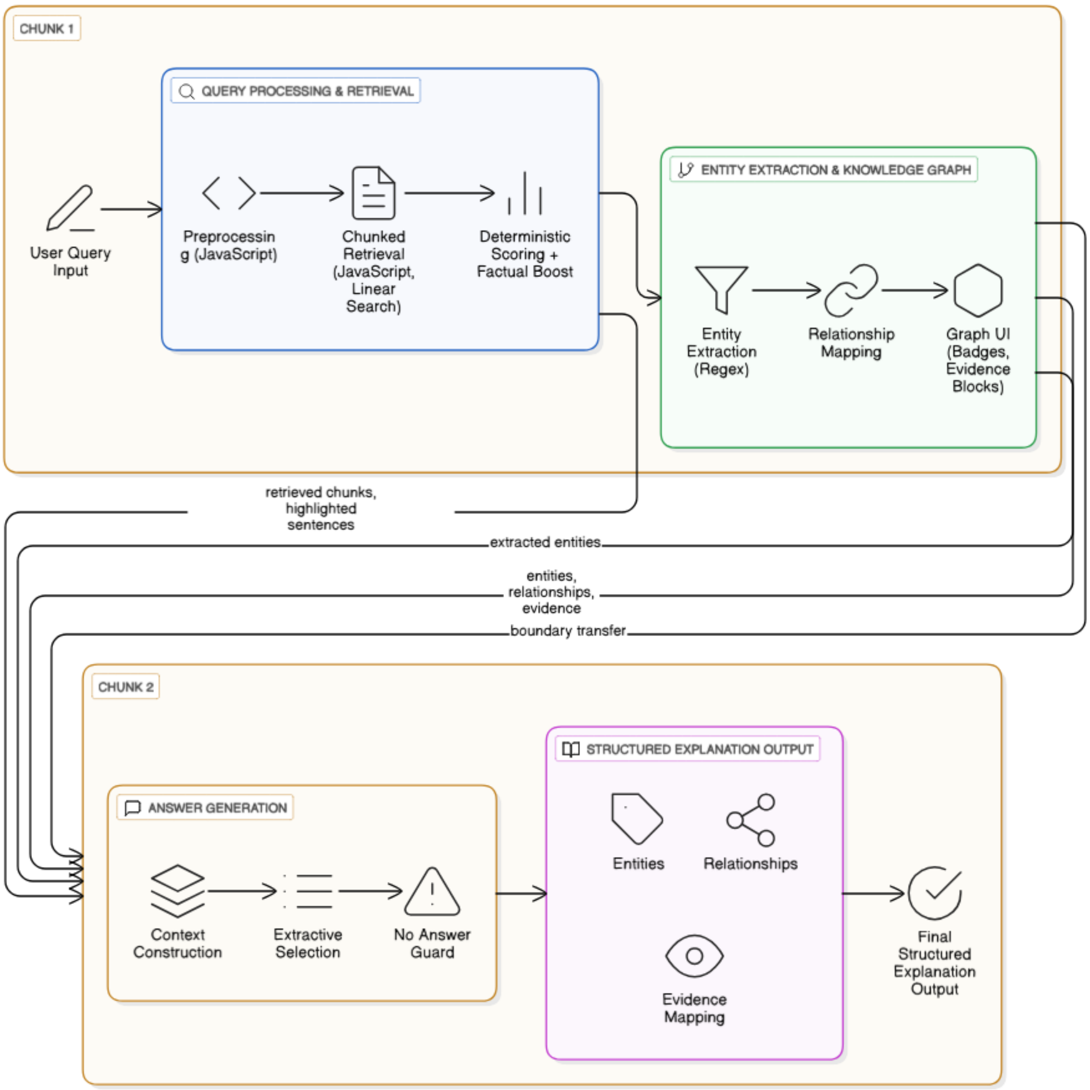

Figure 2: System Overview Architecture. A comprehensive visualization of the four-stage Explainable RAG pipeline, mapping the flow from Query Processing to the final Structured Explanation Output.

Prototype

GitHub Link

Retriever.

Implementation Details

The prototype is a browser-native application that requires no installation. It features a Knowledge Base management system, a real-time search interface, and a tabbed "Answer & Evidence" viewer.

Tech Stack

- **Frontend UI:** HTML5, Tailwind CSS
- **Document Parsing:** `pdf.js` (PDF), `mammoth.browser.js` (DOCX)
- **Processing Logic:** Vanilla JavaScript (ES6+)
- **Retrieval/Graph:** Regex-based extraction and term-frequency scoring
- **Deployment:** Static Web App (Client-side execution)

Demo Example

Query: "What is the economic impact of climate change?"
Output: "The economic cost of climate inaction is estimated at $23 trillion by 2050. Green bonds have reached $500 billion in issuance. Some economists argue the transition costs are underestimated.. Global temperatures have risen by 1.2°C since pre-industrial times."

Future Improvements

While the current prototype establishes a baseline for transparency, the following enhancements are proposed to increase the depth of reasoning and scalability:

- **Multimodal Data Ingestion (OCR):** Integrating Optical Character Recognition (OCR) engines like *Tesseract* or *PaddleOCR* to process scanned documents and images. This would allow the system to extract text from diagrams and flowcharts, which often contain critical process-related information.
- **Structured Table Parsing:** Implementing layout-aware extraction (using tools like `Unstructured.io` or `Camelot`) to convert tables into structured formats like Markdown or JSON. This ensures that the relational data within cells is preserved, allowing the RAG system to answer quantitative queries that require row-column lookups.
- **Hybrid Retrieval:** Integration of semantic vector embeddings (e.g., $SBERT$) alongside the current keyword-based search to capture latent context while maintaining exact-match transparency.
- **SVO Triplet Extraction:** Moving beyond simple co-occurrence to Subject-Verb-Object (SVO) extraction to transform the local graph into a more actionable relational Knowledge Graph.

GitHub Repository

The source code for this prototype is hosted at the following link:
Repo link: Retriever.

Conclusion

This work presents an end-to-end Explainable Retrieval-Augmented Generation (RAG) system that integrates lightweight knowledge graph construction to improve transparency and trust.The X-RAG prototype demonstrates that explainability does not require complex graph databases. By focusing on Local Contextual Extraction and Transparent Scoring, we provide a system where the user can verify every claim. This approach reduces the "hallucination" risk inherent in standard RAG systems and provides a clear audit trail for sensitive data analysis.