RAG-Based AI Research Assistant

Automated Paper Summarization and Analysis

Yousaf Khaliq, Danlei Zhu, Haoyuan Wang

April 28, 2025

Introduction

- A tool to assist researchers doing background work
- Fetches papers from arXiv
- Various similarity methods
- Summarizes papers using Azure OpenAI's GPT-3.5-turbo model
- Chat box to chat with papers
- Web app interface using Gradio

Introduction

1. Background
2. Methodology
    1. PDF Text Extraction Tools
    2. Text Vectorization
    3. Similarity Calculation
    4. RAG Implementation
3. Demonstration & Results
4. Discussion & Conclusions

Current Research

- **Citation Analysis Tools** (e.g., Google Scholar, Scopus): Good for tracking impact, but lack semantic relationships [4].
- **Digital Libraries**: Focus on organizing and accessing literature [9].
- **Traditional Literature Review**: Relies heavily on manual reading (abstract/intro/conclusion), which is time-consuming.
- **Automated Text Summarization**: Useful for quickly assessing relevance [6].
- **Topic Modeling (LDA)**: Helps identify themes and clusters within documents [1].
- **Recent NLP Advances**: Enable more sophisticated semantic analysis [7].

Current Research

- **Large Language Models (LLMs)**:
  - Powerful for automated summarization and synthesis [8, 2].
  - Can be fine-tuned for question answering ("ChatBox" idea) [3].
  - Used for automating article selection in systematic reviews [5].
- **Our Approach**: Leveraging Azure OpenAI GPT-4 for research analysis and assistance.

Existing Tools for Literature Management

Assisting researchers with literature management and understanding is a common goal. Several tool categories exist:

- **Reference Managers (e.g., Zotero, Mendeley):**
  - Excel at collecting, organizing, citing, and managing PDFs.
  - Primary focus: Citation management and organization.
  - Limited semantic querying or deep comparative analysis on full text.

Existing Tools

- **AI-Powered Research Tools (e.g., NotebookLM, Elicit, SciSpace):**
  - Use RAG on uploaded documents (NotebookLM).
  - Extract structured information (Elicit).
  - Offer summarization, matrices (SciSpace).
  - Often closed-source, subscription-based.
  - Focus on user-uploaded or database papers, not specific on-demand fetching.
- **Semantic Search Engines (e.g., Semantic Scholar, Connected Papers):**
  - Focus on discovering relevant literature and visualizing connections (citation/semantic networks).
  - Excellent for discovery.
  - Generally do not offer interactive Q&A or summarization grounded in a *user-defined, just-fetched set*.

PDF Text Extraction Tools

- **pdfplumber**: Accurate layout extraction, excels with structured PDFs. Best for structured layouts, struggles with scanned images.
- **PyMuPDF (fitz)**: High performance, supports Optical Character Recognition (OCR), good compatibility. High speed and versatile, slight formatting issues in complex layouts.
- **PyPDF2**: Lightweight, stable, suitable for simple PDFs. Simple tasks, lacks table/multicolumn support.

Text Vectorization

Term Frequency-Inverse Document Frequency (TF-IDF): Measures word importance using term frequency and inverse document frequency. Fast, no semantic understanding.

$$\text{tfidf}(t, d, D) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}} \times \log \frac{N}{|\{d : t \in d\}|}$$

- t: term (word)
- d: a single document
- D: collection of all documents (corpus)
- $f_{t,d}$: frequency of term t in document d
- N: total number of documents
- $|\{d : t \in d\}|$: number of documents containing term t

Text Vectorization

Latent Semantic Analysis (LSA): Applies SVD for latent semantic structure discovery. Captures topic information via dimensionality reduction.

$$X = U\Sigma V^\top, \quad X_k = U_k \Sigma_k V_k^\top$$

- X: term-document matrix
- U, Σ, V: SVD decomposition matrices
- k: reduced dimension (number of topics)

Text Vectorization

Word2Vec: Neural network-based word embeddings predicting context. Pre-trained embeddings, some semantic awareness.

$$J = \log \sigma(v_{w_o}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n}[\log \sigma(-v_{w_i}^\top v_{w_I})]$$

- J: loss function
- σ: sigmoid function
- v_{w_I}: vector of input word
- v_{w_o}: vector of output word
- w_i: negative sampled word
- P_n: noise distribution

Text Vectorization

BERT: Transformer-based model using bidirectional context for deep semantic understanding. Deep contextual understanding, highest computational cost.

$$\mathcal{L}_{\text{MLM}} = -\mathbb{E}_{x \sim \mathcal{D}} \sum_{i \in M} \log p(x_i | x_{\setminus M})$$

- $\mathcal{L}_{\text{MLM}}$: masked language modeling loss
- x: input sequence
- $\mathcal{D}$: dataset
- M: masked positions
- $p(x_i | x_{\setminus M})$: probability of predicting masked token x_i

Cosine Similarity

- **What is Cosine Similarity?**
  - Measures the similarity between two vectors.
  - Range: $[-1, 1]$, where 1 indicates identical vectors.
- **Mathematical Formula**:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

  where:
  - $\mathbf{A} \cdot \mathbf{B}$: Dot product of vectors $\mathbf{A}$ and $\mathbf{B}$.
  - $\|\mathbf{A}\|$: Euclidean norm of vector $\mathbf{A}$.
- **Application in the Tool**:
  - Compute pairwise cosine similarity to generate a correlation matrix.
  - Visualize the correlation matrix as a heatmap.

Cosine Similarity Comparison

Paper	pdfplumber				pymupdf			
	TF-IDF	LSA	Word2Vec	BERT	TF-IDF	LSA	Word2Vec	BERT
Guo et al.	0.5404	0.5404	0.9121	0.6783	0.5515	0.5515	0.9783	0.8503
Soydaner	0.4722	0.4722	0.9412	0.7974	0.4209	0.4209	0.9699	0.8129
FuKui et al.	0.5384	0.5384	0.9299	0.7649	0.6398	0.6398	0.9777	0.8689
Zhu et al.	0.6690	0.6690	0.9461	0.7922	0.6947	0.6947	0.9581	0.7502
Liu et al.	0.2310	0.2310	0.8442	0.6157	0.4423	0.4423	0.9538	0.7290

PdfReader				
Paper	TF-IDF	LSA	Word2Vec	BERT
Guo et al.	0.6112	0.6112	0.9794	0.8097
Soydaner	0.4244	0.4244	0.9682	0.8380
FuKui et al.	0.2015	0.2015	0.8073	0.8646
Zhu et al.	0.7311	0.7311	0.9721	0.7539
Liu et al.	0.4614	0.4614	0.9494	0.7554

What is Retrieval-Augmented Generation (RAG)?

The Challenge with Standard LLMs:

- **Knowledge Cutoff:** Pre-trained models have knowledge limited to their training data (can be outdated).
- **Hallucination:** Can generate plausible but factually incorrect or nonsensical information.
- **Lack of Grounding:** Answers aren't directly tied to specific, verifiable sources by default.

RAG: Combining Strengths

- RAG addresses these limitations by integrating information retrieval with large language model generation.
- **Goal:** Ground LLM responses in factual information retrieved from a specific, trusted knowledge corpus.

Core Components

1. **Retriever:**
   - Takes the user query.
   - Searches a knowledge corpus (e.g., documents, database) for relevant information snippets.
   - Common techniques involve vector embeddings and similarity search (e.g., using FAISS).
2. **Generator:**
   - Typically a Large Language Model (LLM) like GPT.
   - Receives the original query AND the retrieved context.
   - Is prompted to generate an answer based *only* on the provided context.

General Workflow

Query $\rightarrow$ Retriever finds relevant docs $\rightarrow$ Docs + Query fed to LLM $\rightarrow$ LLM generates grounded answer.

Benefits: Increased accuracy, reduced hallucination, up-to-date information (if corpus is current), and potential for source citation.

RAG Implementation in Our AI Assistant

Specific Goal: Answer user questions (summaries, comparisons) about a set of arXiv papers fetched on-demand for a specific topic.

Implementation Details:

- **Knowledge Corpus:** The set of PDF documents downloaded from arXiv based on the user's topic query during the current session.
- **Indexing Phase** (`load_and_index_papers_for_rag`):
  - *Loader:* LangChain `PyPDFLoader` (extracts text page-by-page).
  - *Splitting:* LangChain `RecursiveCharacterTextSplitter` (creates manageable chunks).
  - *Embedding:* LangChain `OpenAIEmbeddings` (using `text-embedding-ada-002` or similar via API).
  - *Vector Store:* `FAISS` (CPU version for efficient local similarity search). Metadata includes source filepath, title, page number (`'page'`).

Retrieval Phase

- *Parsing:* Identifies target paper indices/titles from user query (`parse_paper_references`).
- *Retriever:* FAISS `similarity_search` is used.
- *Filtering:* Applies metadata filter (`'source': filepath`) to isolate chunks from specific target papers.
- *Query Strategy:* Uses paper title, then original question, then generic query as fallbacks during filtered retrieval.

Generation Phase

- *LLM:* LangChain `ChatOpenAI` (using `gpt-3.5-turbo` model via API).
- *Chain:* LangChain `load_qa_chain` with `chain_type="stuff"`.
- *Prompting:*
  - Main QA Prompt (`qa_prompt`): Instructs LLM on task, constraints (use context only), and desired output format. Includes post-context instruction.
  - Document Prompt (`document_prompt`): Formats each retrieved chunk passed to the LLM, adding explicit source markers (`**Source: 'Title' - page X**...`) to aid attribution, especially for comparisons.
- *Question Modification:* For single-paper summary requests via index, the question passed to the LLM is rephrased for clarity (e.g., "Summarize the key points... from the paper titled 'X'").

Output Processing

- Checks for LLM refusals based on keywords/answer length.
- Prepends paper title(s) to the response if queried by index.
- Appends formatted sources (`'Title', page X`) *only* if the LLM provided a non-refusal answer.

$\Longrightarrow$ This provides answers grounded specifically in the fetched arXiv papers for the current session.

Discussion: Limitations and Challenges

Data Input and Processing:

- PDF Text Extraction Quality
- RAG Chunking & Retrieval

LLM (GPT-3.5 Turbo) Behavior:

- Validation of responses
- Summarization/Comparison Nuance
- Occasional Confusion
- **S**trict Context Adherence

Discussion Cont.

Similarity Metric (TF-IDF):

- Measures *lexical* (term-based) similarity, not deep semantic meaning
- Sensitive to the text cleaning process

Scalability and Cost:

- Processing many large PDFs
- Relies on OpenAI API calls for embeddings and generation

Conclusion and Future Directions

Summary:

- We successfully developed an AI Research Assistant integrating on-demand arXiv paper fetching, Retrieval-Augmented Generation (RAG) for interactive Q&A, and TF-IDF based similarity visualization.

Key Contributions:

- **Grounded Q&A:** Enables users to summarize and compare papers fetched within the current session, with answers grounded in the source text and including source citations.
- **Integrated Workflow:** Combines discovery (fetching), understanding (RAG chat), and relationship exploration (similarity plots) in one tool.
- **Transparency & Adaptability:** Demonstrates a practical RAG implementation using standard libraries, offering potential for modification.

Conclusions Cont.

Value Proposition:

- Potential to significantly accelerate the initial stages of literature review by automating fetching and providing tools for rapid content analysis and comparison.
- Aids comprehension and helps identify potential connections between research papers.

Limitations Acknowledged:

- Performance is contingent on PDF quality and text extraction success.
- RAG retrieval and LLM generation (GPT-3.5) have inherent nuances and limitations.
- TF-IDF provides only a surface-level textual similarity view.

Future Work

- Integrate more robust PDF parsing libraries.
- Experiment with more advanced LLMs (e.g., GPT-4/o) for higher-quality generation.
- Explore alternative retrieval strategies (e.g., HyDE, re-ranking).
- Incorporate different similarity metrics (e.g., semantic embedding similarity, citation analysis via external APIs).
- Enhance UI/UX, including better error handling and plot labeling.

DM Blei, AY Ng, and MI Jordan.
Latent dirichlet allocation, journal of machine learning research 3 (jan).
2003.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al.
Language models are few-shot learners.
Advances in neural information processing systems, 33:1877–1901, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
Bert: Pre-training of deep bidirectional transformers for language understanding.
In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language*

technologies, volume 1 (long and short papers), pages 4171–4186, 2019.

Eugene Garfield.

Citation indexes for science: A new dimension in documentation through association of ideas.

Science, 122(3159):108–111, 1955.

Seyed Mohammad Ali Jafari.

Streamlining the selection phase of systematic literature reviews (slrs) using ai-enabled gpt-4 assistant api.

arXiv preprint arXiv:2402.18582, 2024.

Inderjeet Mani and Mark T Maybury.

Advances in automatic text summarization.

MIT press, 1999.

Christopher Manning and Hinrich Schutze.

Foundations of statistical natural language processing.

MIT press, 1999.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin.

Attention is all you need.

Advances in neural information processing systems, 30, 2017.

Ian H Witten, David Bainbridge, and David M Nichols.

How to build a digital library.

Morgan Kaufmann, 2009.