Indian Institute Of Technology, Kanpur

Automated Extraction of Synthesis Data from Unstructured Scientific Literature

Multi-Modal and Multi-Pipeline Retrieval-Augmentation Generation

Author
Arsh Jaswal

Mentor
Prof. Salman Ahmad Khan

Date: 02/07/2025

CERTIFICATE

This is to certify that the project entitled ' Automated extraction of synthesis data from unstructured scientific literature ' submitted by Arsh Jaswal as a part of Students-Undergraduate Research Graduate Excellence 2025 offered by the Indian Institute of Technology, Kanpur, is a bonafide record of the work done by him/her under my guidance and supervision at the Indian Institute of Technology, Kanpur.

Mentor's Name: Salman Ahmad Khan
Department: Chemical Engineering
Indian Institute of Technology, Kanpur

[illegible]
Mentor's Signature

Abstract

The volume of scientific literature on chemical and material synthesis is expanding rapidly, yet much of this information remains embedded in unstructured text, hindering its accessibility and utility. The work presents a natural language processing (NLP) and large language model (LLM)-based pipeline for extracting, organizing, and querying synthesis data from unstructured journal articles over the web. As a case study, we focus on pulsed laser deposition (PLD), a widely used thin-film synthesis technique. The pipeline involves scraping scientific articles, identifying relevant PLD synthesis sections using ModernBERT contextual embeddings, and extracting key synthesis parameters, such as temperature, pressure, and laser intensity, using state-of-the-art LLMs, including LLaMa and Gemini. Comparative analysis of these models assesses their effectiveness in extracting technical data from scientific text. We implement a Retrieval-Augmented Generation (RAG) system integrated with a multi-retrieval and cloud-based vector database architecture to support a chatbot capable of responding to user queries on PLD synthesis process or its conditions. This framework not only facilitates streamlined access to synthesis protocols for experimentalists but also serves as a scalable and adaptable tool for broader applications across different research domains.

Content

Introduction

Pulsed Laser Deposition (PLD) is a widely adopted physical vapor deposition technique for fabricating thin films, valued for its precision and versatility. The process relies on finely tuned experimental parameters, typically sourced through manual review of extensive scientific literature, a task that is both time-consuming and inefficient.

Traditional keyword-based search methods are often inefficient for extracting specific experimental setups, such as synthesis parameters. In this context, Natural Language Processing (NLP) techniques, such as Retrieval-Augmented Generation (RAG), offer promising solutions by combining document retrieval with large language models to generate context-aware responses [1]. It works in three stages: retrieval, augmentation of the LLM's input with the retrieved content, and generation of a response grounded in both retrieved and pre-trained knowledge.

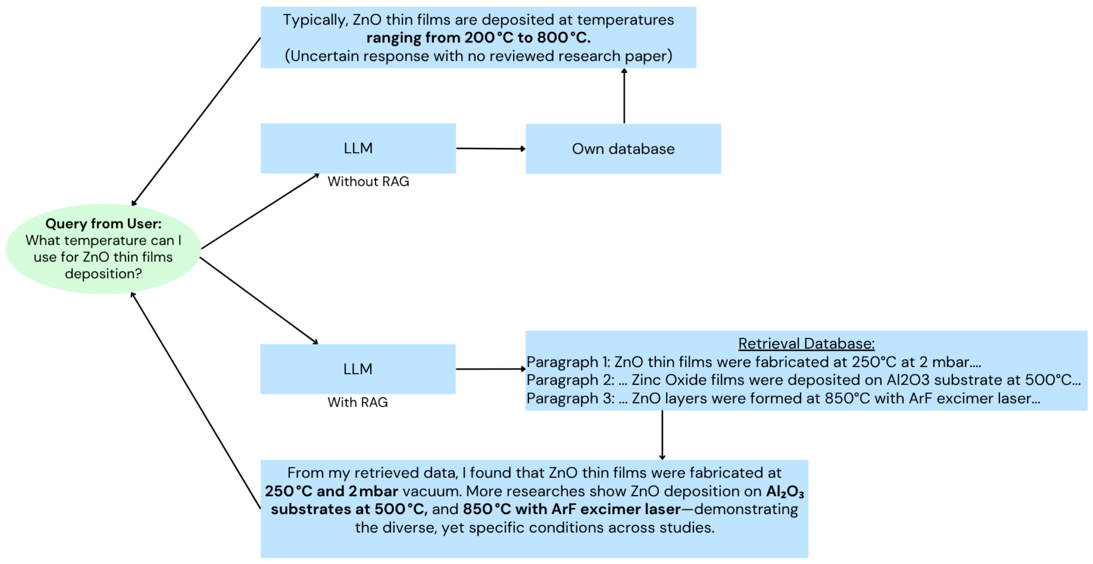

Typically, ZnO thin films are deposited at temperatures **ranging from 200 °C to 800 °C.** (Uncertain response with no reviewed research paper)
LLM
Own database
Without RAG
Query from User: What temperature can I use for ZnO thin films deposition?
LLM
With RAG
Retrieval Database:
Paragraph 1: ZnO thin films were fabricated at 250°C at 2 mbar....
Paragraph 2: ... Zinc Oxide films were deposited on Al2O3 substrate at 500°C...
Paragraph 3: ... ZnO layers were formed at 850°C with ArF excimer laser...
From my retrieved data, I found that ZnO thin films were fabricated at **250 °C and 2 mbar** vacuum. More researches show ZnO deposition on **Al_2O_3 substrates at 500 °C,** and **850 °C with ArF excimer laser**—demonstrating the diverse, yet specific conditions across studies.

Figure 1: Fusion of Retrieval-Augmented Generation with PLD literature

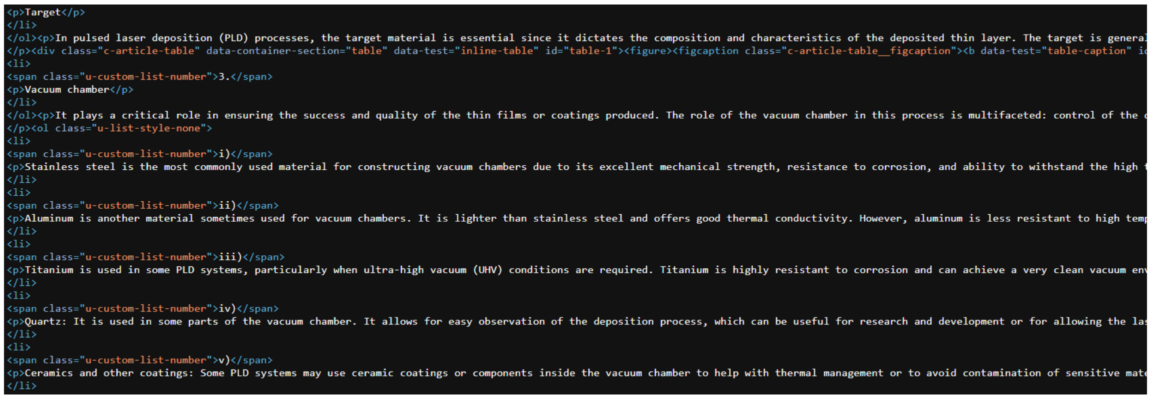
We develop a synthesis-specific paragraph database from Springer articles by parsing HTML with BeautifulSoup4 and applying supervised classification, using PLD as a case study. Retrieval-Augmented Generation (RAG) leverages semantic similarity via contextual embeddings and cosine distance to improve query-to-document alignment. However, retrieval inaccuracies can lead to erroneous or hallucinated outputs, underscoring the need for enhanced retrieval precision to ensure reliable information extraction in domains with high semantic overlap.

Extracting Synthesis Paragraphs

Scientific research articles available online are typically found in three formats: HTML, PDF, or JSON. All three formats are widely used, but the PDF format is the most challenging to process due to its LaTeX-based layout, inconsistent formatting, and lack of a structured, accessible code representation. On the other hand, HTML and JSON formats are more structured, making them significantly easier to parse for content extraction [2]. In this work, we focus on HTML-based extraction using a combination of API access and direct web scraping, followed by paragraph classification to support a retrieval-based NLP pipeline.

To extract paragraphs from HTML pages, we utilized BeautifulSoup4, a widely used Python library for parsing HTML and XML documents. BeautifulSoup4 allows developers to navigate, search, and modify parse trees efficiently. It is particularly effective at locating specific HTML tags, e.g., `<p>, <div>, <section>`, and content structures, making it suitable for extracting textual data from scholarly articles.

We used the Springer Nature Metadata and Full-Text API (`https://dev.springernature.com`) for automated access to full texts [3]. Although structured, API content was often less complete than direct HTML scraping, prompting us to develop a generalized web scraper for cleaner extraction. All scraping complied with site policies, limiting data to 50 papers for demonstration. For large-scale scraping, batch processing is recommended to minimize server load and respect publisher guidelines.



```
<p>Target</p>
</li>
</ol><p>In pulsed laser deposition (PLD) processes, the target material is essential since it dictates the composition and characteristics of the deposited thin layer. The target is genera
</p><div class="c-article-table" data-container-section="table" data-test="inline-table" id="table-1"><figure><figcaption class="c-article-table__figcaption"><b data-test="table-caption" i
<li>
<span class="u-custom-list-number">3.</span>
<p>Vacuum chamber</p>
</li>
</ol><p>It plays a critical role in ensuring the success and quality of the thin films or coatings produced. The role of the vacuum chamber in this process is multifaceted: control of the
</p><ol class="u-list-style-none">
<li>
<span class="u-custom-list-number">i)</span>
<p>Stainless steel is the most commonly used material for constructing vacuum chambers due to its excellent mechanical strength, resistance to corrosion, and ability to withstand the high
</li>
<li>
<span class="u-custom-list-number">ii)</span>
<p>Aluminum is another material sometimes used for vacuum chambers. It is lighter than stainless steel and offers good thermal conductivity. However, aluminum is less resistant to high tem
</li>
<li>
<span class="u-custom-list-number">iii)</span>
<p>Titanium is used in some PLD systems, particularly when ultra-high vacuum (UHV) conditions are required. Titanium is highly resistant to corrosion and can achieve a very clean vacuum en
</li>
<li>
<span class="u-custom-list-number">iv)</span>
<p>Quartz: It is used in some parts of the vacuum chamber. It allows for easy observation of the deposition process, which can be useful for research and development or for allowing the la
</li>
<li>
<span class="u-custom-list-number">v)</span>
<p>Ceramics and other coatings: Some PLD systems may use ceramic coatings or components inside the vacuum chamber to help with thermal management or to avoid contamination of sensitive mat
</li>
```

Figure 2: `<p> tags in the extracted html paper`

The extraction pipeline focused on identifying the `main-content` section of each article's HTML structure. This section typically contains the body of the paper, where paragraphs are cleanly marked with `<p>` tags. Using this structure, we were able to efficiently extract paragraphs while discarding non-essential elements such as headers, figures, references, and metadata.

From this process, we extracted approximately 4,000 paragraphs. The primary task that followed was to classify each paragraph $p_i \in \mathcal{P}$ into one of two classes: synthesis ($y_i = 1$) and non-synthesis ($y_i = 0$), where $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ is the set of extracted paragraphs. The classification task can be defined as a mapping function:

$$f : \mathcal{P} \to \{0, 1\}$$

trained to maximize prediction accuracy on a highly imbalanced dataset. The aim was to create a high-quality database consisting only of synthesis-related content, which is crucial for downstream information retrieval tasks in domains such as material science and chemical engineering.

We manually labeled 2,000 paragraphs for training, with only about 100 being synthesis-related, revealing a notable class imbalance. A baseline logistic regression model using TF-IDF features achieved an F1 score of 0.75 for the minority class. To address this imbalance, we augmented the dataset with approximately 1,000 additional synthesis paragraphs from an external source [3], which increased the F1 score to 0.96 and overall accuracy to 98%.

We also evaluated alternative models and features, including BERT and SciBERT embeddings, support vector machines (SVM), principal component analysis (PCA), and random forests. Most did not surpass the TF-IDF logistic regression baseline. Despite SVM with TF-IDF being competitive, it underperformed compared to logistic regression. Given the simplicity and efficiency of the logistic regression model, achieving an AUC-ROC of 0.9974; we retained this configuration. Combined with a web scraper and HTML parser, this pipeline enables precise extraction of synthesis-specific paragraphs, supporting a retrieval-augmented system for domain-specific query answering.

Constructing the Retrieval Infrastructure

In our Retrieval-Augmented Generation (RAG) pipeline, a user query $x \in \mathcal{X}$ is processed by a retriever model R, which selects the top-k paragraphs $D = \{d_1, d_2, \ldots, d_k\} \subset \mathcal{P}$ from a corpus $\mathcal{P}$ of 9000+ paragraphs. These documents are then passed to a generator G, which generates an output $y \in \mathcal{Y}$ conditioned on both x and D.

Formally, the system models the conditional distribution:

$$P(y \mid x) = \sum_{d \in D} P_\theta(y \mid x, d) \cdot P_\phi(d \mid x)$$

where $P_\phi(d \mid x)$ is the retriever's relevance score and $P_\theta(y \mid x, d)$ is the likelihood under the generator. These paragraphs, although topically relevant, were semantically dense and exhibited high inter-similarity due to repeated mentions of similar experimental setups, materials, and parameter configurations. This created a significant challenge for semantic search, as traditional embedding-based retrieval often failed to differentiate between meaningfully distinct paragraphs.

Initial tests with specialized transformers (MatSciBERT, SciBERT) underperformed, likely because their embeddings missed subtle context in experimental texts. Even techniques like chunking [1] provided only marginal improvements. We adopted ModernBERT for retrieval in highly repetitive scientific domains due to its superior sentence-level semantic discrimination. Each paragraph $p_i \in \mathcal{P}$ and the user query $x \in \mathcal{X}$ are embedded into a shared vector space using a contextual encoder function $\mathrm{Enc}(\cdot)$, such that:

$$\mathbf{v}_x = \mathrm{Enc}(x), \quad \mathbf{v}_{p_i} = \mathrm{Enc}(p_i)$$

where $\mathbf{v}_x \in R^d$ and $\mathbf{v}_{p_i} \in R^d$ are the resulting dense vector embeddings of the query and paragraph, respectively.

The retriever then selects the top-k most relevant paragraphs by ranking them according to their similarity scores:

$$D = \mathrm{TopK}_{p_i \in \mathcal{P}} \left(\mathrm{sim}(x, p_i)\right)$$

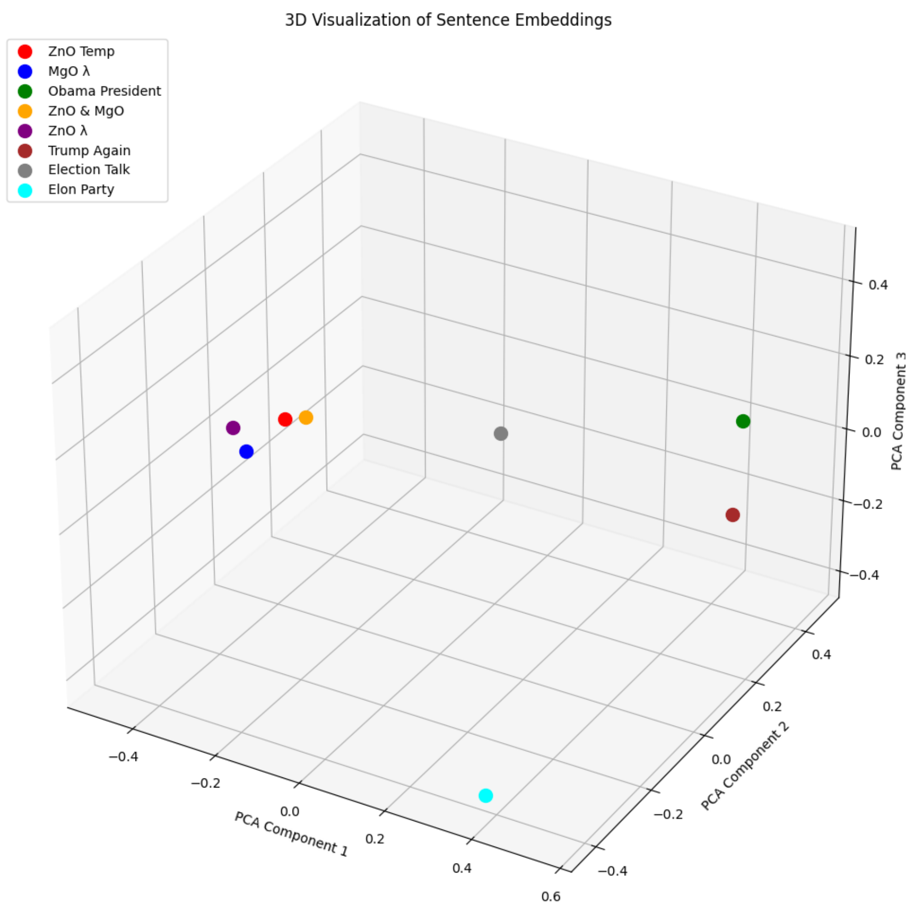
where $D = \{d_1, d_2, \ldots, d_k\} \subset \mathcal{P}$ is the set of top-k retrieved paragraphs used in the downstream generation phase, ranked such that

$$\mathrm{sim}(x, d_1) > \mathrm{sim}(x, d_2) > \cdots > \mathrm{sim}(x, d_k),$$

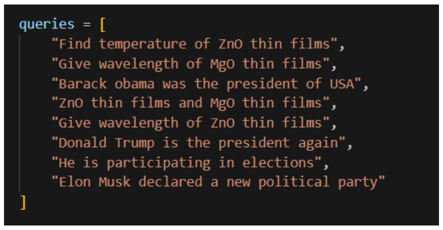
where

$$\mathrm{sim}(x, d_i) = \frac{\langle f(x), f(d_i) \rangle}{\|f(x)\| \cdot \|f(d_i)\|},$$

and $f : \mathcal{X} \cup \mathcal{P} \to R^n$ is the embedding function, $\mathcal{X}$: set of queries; $\mathcal{P}$: set of paragraphs; n: dimension of the numeric vector. The spatial distribution of such paragraphs is illustrated in Figure 3.



(a) 3D representation after dimensionality reduction (n=3)



```
queries = [
    "Find temperature of ZnO thin films",
    "Give wavelength of MgO thin films",
    "Barack obama was the president of USA",
    "ZnO thin films and MgO thin films",
    "Give wavelength of ZnO thin films",
    "Donald Trump is the president again",
    "He is participating in elections",
    "Elon Musk declared a new political party"
]
```

(b) Queries used for the plot

Figure 3: 3D representations of sentence embeddings

Semantic similarity alone was insufficient for optimal retrieval, as large language models (LLMs) often struggle with unstructured queries lacking contextual cues. To address this, we applied query expansion, reformulating the input query x into a rewritten query x' by adding domain-specific keywords, removing irrelevant helping verbs, and incorporating terms derived from corpus co-occurrence statistics. This improved alignment between x' and corpus vocabulary, enhancing retrieval of relevant but less obvious content [4].

Despite gains from ModernBERT embeddings and query expansion, retrieval occasionally returned irrelevant or redundant paragraphs due to dense semantic overlap. To refine results, we enhanced retrieval process [4, 5] by developing a multi-stage reranking pipeline, Deep Search, composed of:

Filter Pruning: Retrieve the top 200 paragraphs $\{p_i\}$ ranked by cosine similarity $\text{sim}(x', p_i)$. Each paragraph receives a keyword score

$$\text{kw\_score}(p_i) = \sum_{k \in K(x')} \mathbf{1}_{k \in p_i} \quad \text{where} \quad \mathbf{1}_{k \in p_i} = \begin{cases} 1 & \text{if keyword } k \text{ is in } p_i \\ 0 & \text{otherwise} \end{cases}$$

with $K(x')$ denoting keywords extracted from x'. Paragraphs are sorted by kw_score in descending order, and the top 50 are retained.

Heuristic Reranking: Combines lexical and semantic signals to further refine the filtered paragraphs.

Multi-Semantic Matching: Conducts a final semantic similarity search to robustly select the top-k paragraphs.

All embeddings are stored and retrieved via the Pinecone vector database, enabling scalable, low-latency vector search. Evaluation showed Deep Search significantly outperformed baseline models in precision and relevance, with domain experts confirming improved alignment to experimental intent. Though computational overhead increased, the gain in retrieval quality justifies the additional processing time in precision-critical research applications.

Designing the Generation Pipelines

To support accurate, efficient, and modular language generation, we designed and evaluated multiple generation pipelines. Each pipeline was tailored for specific trade-offs in latency, complexity, and output quality, and all were built to be adaptable across various LLM backends [1].

The first pipeline utilized **semantic search with query rewriting** [5], optimized for low-latency applications. Users can define a top-k parameter to control how many semantically similar paragraphs are retrieved. These passages are then summarized to retain essential details (e.g., numerical data, parameters) while ensuring the input remains within the model's context limit.
The generation process operates as a **multi-pass transformation**, where:

- Let P represent the summarized paragraph set.
- The first generation step, $G_1 : P \to A$, generates a raw answer A.
- The second generation, $G_2 : A \to A^*$, refines A into a more coherent, fluent, and professional response A^*, optionally integrating relevant external knowledge.

This pipeline is lightweight and ideal for fast-response settings where acceptable quality is sufficient.

The Deep Search pipeline extended the semantic-only model by incorporating more thorough multi-layer retrieval mechanisms. While the underlying logic remained similar, this version emphasized deeper understanding and precision. It still followed the G_1 and G_2 structure but significantly improved output quality at the expense of processing time. This made it suitable for complex or high-stakes queries.

To support multi-turn conversations, we introduced **conversation history tracking** in the deep search pipeline. This version maintains the last five interactions in a fixed-size queue Q:

$$Q = [t_1, t_2, \ldots, t_5]$$

When a new turn t_{new} is added:

$$Q = \begin{cases} Q \cup \{t_{\text{new}}\}, & \text{if } |Q| < 5 \\ (Q[2:], t_{\text{new}}), & \text{otherwise} \end{cases}$$

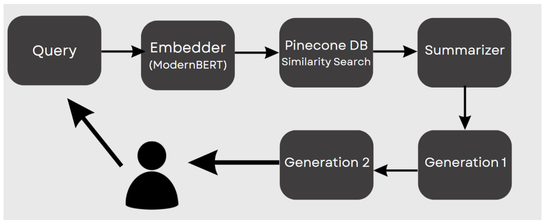
This ensures context continuity without exceeding the model's token window, enhancing coherence and avoiding redundancy in multi-turn scenarios.

A distinct pipeline was implemented for **data visualization**, specifically histogram plotting. This pipeline is useful in scenarios where users require analytical insights in addition to text generation. The process includes:

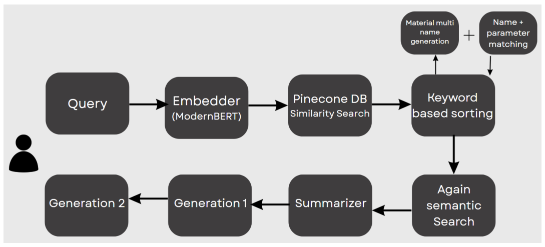
- **Extraction:** Numerical values are identified and extracted from semantically retrieved paragraphs.

- **Preprocessing:** The data is normalized, filtered, or grouped based on context (e.g., unit standardization, rounding).
- **Plotting:** The cleaned data is visualized using histograms. These plots help users identify patterns such as frequency distributions, anomalies, or trends over categories.

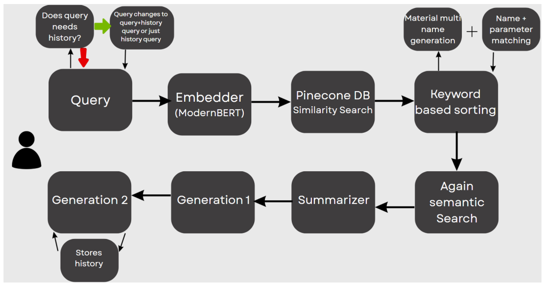
This visual pipeline expands the model's utility in research, reporting, and decision-making environments where graphical output is more actionable than text.



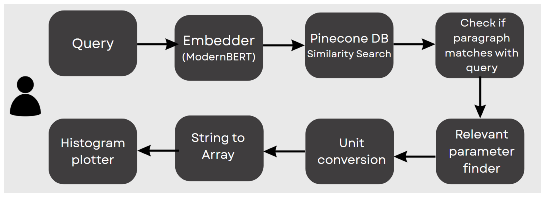
(a) Flow chart of semantic search only pipeline



(b) Flow chart of deep search pipeline


Does query needs history?
Query changes to query+history query or just history query
Material multi name generation
+
Name + parameter matching
Query
Embedder (ModernBERT)
Pinecone DB Similarity Search
Keyword based sorting
Again semantic Search
Summarizer
Generation 1
Generation 2
Stores history

(c) Flow chart of deep search with history pipeline


Query
Embedder (ModernBERT)
Pinecone DB Similarity Search
Check if paragraph matches with query
Relevant parameter finder
Unit conversion
String to Array
Histogram plotter

(d) Flow chart of histogram plotter pipeline

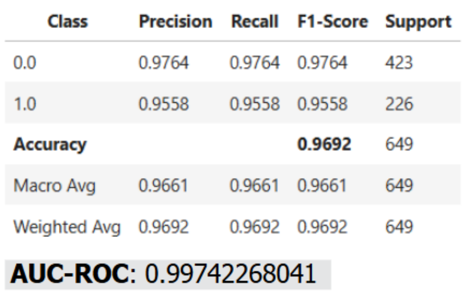
Figure 4: Flowchart of pipelines

We evaluated multiple LLMs: *Gemini 2.0 Flash*, *Phi-3 Mini*, *LLaMA 3.2 8B*, and *DeepSeek*. Ultimately, **Gemini 1.5 Flash** was selected due to its fast API, lightweight design, and strong contextual accuracy. The semantic search-only pipeline is already deployed as a web app. All pipelines are built with flexibility in mind, using the *Pinecone API* for seamless document scaling without added latency.

API access is available at: `https://aistudio.google.com/welcome`

Results

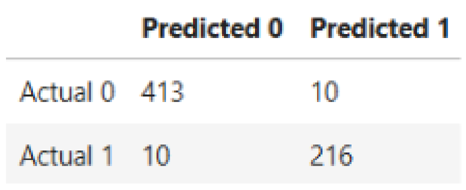
The classifier demonstrated strong performance, as shown in the classification report and confusion matrix below:



Class	Precision	Recall	F1-Score	Support
0.0	0.9764	0.9764	0.9764	423
1.0	0.9558	0.9558	0.9558	226
Accuracy			**0.9692**	649
Macro Avg	0.9661	0.9661	0.9661	649
Weighted Avg	0.9692	0.9692	0.9692	649

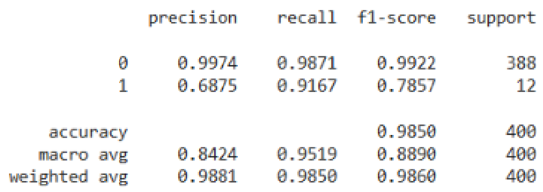
AUC-ROC: 0.99742268041

(a) Classification report after augmentation



	Predicted 0	Predicted 1
Actual 0	413	10
Actual 1	10	216

(b) Confusion matrix after augmentation



```
              precision    recall  f1-score   support

           0     0.9974    0.9871    0.9922       388
           1     0.6875    0.9167    0.7857        12

    accuracy                         0.9850       400
   macro avg     0.8424    0.9519    0.8890       400
weighted avg     0.9881    0.9850    0.9860       400
```

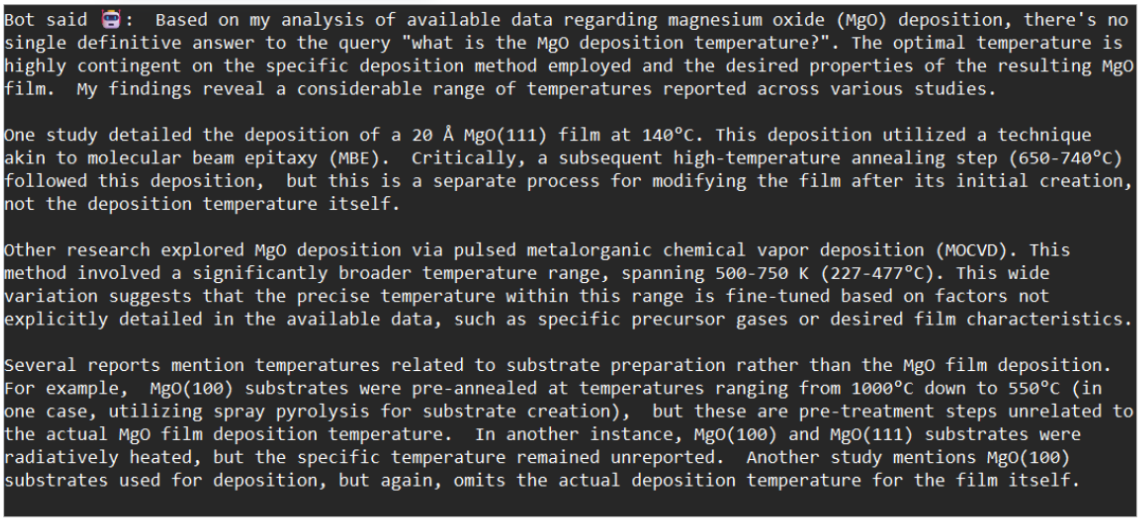
(c) Classification report before augmentation

Figure 5: Results of paragraph extraction process

In addition, the HTML parser and scraper were effectively generalized to extract paragraph-level content from diverse web sources. This preprocessing step ensured consistent and clean input for downstream tasks such as retrieval and classification, contributing to the overall robustness of the system.

This study also evaluates semantic and deep search strategies on a dataset comprising over 9,000 paragraphs. The semantic search system was deployed as a web application using Pinecone, which enabled fast and scalable vector retrieval, averaging approximately 15 to 20 seconds per query with top-$k = 10$. Pinecone allowed seamless data expansion without requiring code modifications or introducing significant latency. While semantic search yielded decent results, its primary advantage was speed and deployment simplicity.

In contrast, the deep search approach produced more accurate and context-aware outputs (Figure 6.), with moderate query times around 35 to 40 seconds per query for top-$k = 10$. It successfully retrieved rare entries, such as the only paragraph on Zr-based materials, and effectively clustered all relevant paragraphs from hundreds on ZnO deposition. Key enhancements included heuristic reranking and filter pruning, which significantly improved result precision and reduced irrelevant outputs. The integration of history tracking further improved coherence and relevance in multi-turn queries.


```
Bot said :  Based on my analysis of available data regarding magnesium oxide (MgO) deposition, there's no
single definitive answer to the query "what is the MgO deposition temperature?". The optimal temperature is
highly contingent on the specific deposition method employed and the desired properties of the resulting MgO
film.  My findings reveal a considerable range of temperatures reported across various studies.

One study detailed the deposition of a 20 Å MgO(111) film at 140°C. This deposition utilized a technique
akin to molecular beam epitaxy (MBE).  Critically, a subsequent high-temperature annealing step (650-740°C)
followed this deposition,  but this is a separate process for modifying the film after its initial creation,
not the deposition temperature itself.

Other research explored MgO deposition via pulsed metalorganic chemical vapor deposition (MOCVD). This
method involved a significantly broader temperature range, spanning 500-750 K (227-477°C). This wide
variation suggests that the precise temperature within this range is fine-tuned based on factors not
explicitly detailed in the available data, such as specific precursor gases or desired film characteristics.

Several reports mention temperatures related to substrate preparation rather than the MgO film deposition.
For example,  MgO(100) substrates were pre-annealed at temperatures ranging from 1000°C down to 550°C (in
one case, utilizing spray pyrolysis for substrate creation),  but these are pre-treatment steps unrelated to
the actual MgO film deposition temperature.  In another instance, MgO(100) and MgO(111) substrates were
radiatively heated, but the specific temperature remained unreported.  Another study mentions MgO(100)
substrates used for deposition, but again, omits the actual deposition temperature for the film itself.
```

Figure 6: Deep Search Result for Query: `"What is the MgO deposition temperature?"` (k=10)

Histogram plotter (Figure 7.) was manually validated to check the accuracy and trend. Although effective, it experienced some latency due to redundant checks. Future improvements may include pipeline optimization, caching mechanisms, or integration with a Model Context Protocol (MCP).

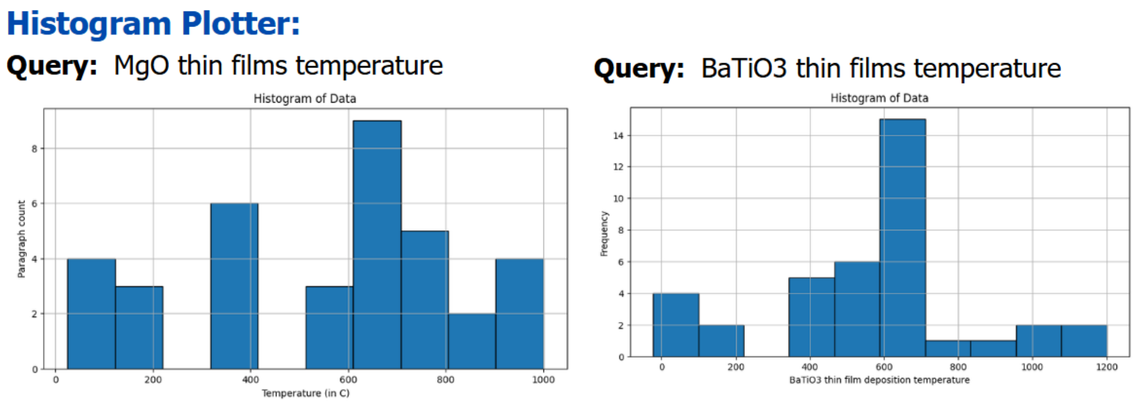


Figure 7: Some examples of Histogram plotter

Overall, deep search offered superior contextual performance, while semantic search provided speed and scalability. Prompt engineering remains a promising area for further enhancement.

Conclusion

This work successfully developed and evaluated a multi-modal, multi-pipeline Retrieval-Augmented Generation (RAG) system for extracting and querying synthesis data from unstructured scientific literature, specifically focusing on Pulsed Laser Deposition (PLD). The system achieved high accuracy in paragraph classification (98% with an F1 score of 0.96 for the minority class) and demonstrated that a pipeline with enhanced retrieval techniques like heuristic reranking, multi-semantic and filter pruning, called as "Deep Search" pipeline significantly outperforms a "Semantic Search" only approach in contextual accuracy, though at a moderate increase in query time. The integration of history tracking, query expansion, and a histogram plotter further enhanced the system's utility, creating a robust and versatile tool for experimentalists and researchers. This framework not only facilitates streamlined access to synthesis protocols for experimentalists but also serves as a scalable and adaptable tool for broader applications across different research domains.

Code can be found here: `https://github.com/arshcwb/researchRAG`

References

1. https://arxiv.org/pdf/2005.11401
2. https://doi.org/10.1038/s41467-024-45563-x
3. https://doi.org/10.1039/D4DD00051J
4. https://arxiv.org/html/2410.19572