



## **AI Patent Advisor: Leveraging LLMs for Patent Analysis & Technology Transfer**

**Report Presented to**

**Prof. DEEPAK SUBRAMANI**

**&**

**Prof. SASHIKUMAR GANESHAN**

**Department of Computational and Data Science**

**Indian Institute of Science, Bangalore**

**On July 26<sup>th</sup>, 2025**

**Course - Post graduate Certificate in AI and MLOPS**

**Cohort 4 – Group 12**

### **Team Members**

Aditya Adhikary,

Aniket Rastogi,

Anbarasan Vellaiyan,

Chaitra HS,

Chandan Srivastava,

Elizebath Thomas,

Kalyan,

Piyush Naik,

Prabhakaran Sambasivam,

Sayan Chowdhury

### **Mentor**

Manohar

Palanisamy

# Table of Contents

[Introduction](#)

[Problem Statement](#)

[Key Problems Addressed](#)

[Overall Objectives](#)

[Journey & Challenges](#)

[Identifying the Dataset](#)

[Settling on Harvard USPTO Patent Dataset](#)

[Preprocessing and Exploratory Data Analysis](#)

[Challenges Choosing between Fine Tuning and RAG](#)

[Challenges Choosing the Right RAG Technique](#)

[Challenges Building Knowledge Graph](#)

[LightRAG – Detailed Explanation](#)

[Building the Chatbot User Interface](#)

[ChatBot System Architecture](#)

[Monitoring & Evaluation](#)

[Key Metrics Tracked](#)

[Monitoring Features](#)

[Evaluation Summary](#)

[Additional Features in the Pipeline](#)

[\[Feature\] Engineering User Input For Improving Context](#)

[\[Feature\] Fallback to Direct Queries to Neo4j In case of LightRAG errors](#)

[\[Feature\] Guardrails](#)

[Doc Structure](#)

[Screenshots](#)

[Patent Insertion Pipeline](#)

[LightRAG - Default Chat UI](#)

[Patent Chatbot Interface](#)

[Neo4j \(Neo4J Aura\) - Graph Visualizations](#)

[VectorDB \(Postgres Serverless - Neon\)](#)

[Monitoring System](#)

[Appendix](#)

[References](#)

# Introduction

Patent documents are a rich source of technological innovation, legal protection, and commercial opportunity. However, the rapid expansion of patent databases—often exceeding millions of entries—makes it increasingly difficult for businesses, legal professionals, and researchers to identify relevant insights and opportunities. Patent analysis and search is crucial for technology scouting, competitive intelligence, and legal risk assessment. Although common, traditional keyword-based search systems often return noisy or irrelevant results. Patent datasets are vast, domain-specific, and inherently difficult to search effectively. Users often encounter synonym mismatches, lack of context, and poorly ranked outputs, leading to suboptimal insights.

This project introduces an AI-powered solution for patent analysis, utilizing a powerful, cutting edge **Retrieval-Augmented Generation (LightRAG) pipeline**. Our pipeline supports the ingestion and semantic processing of structured patent data, embedding it into a **vector database and additionally a graph knowledge database** for efficient and highly accurate context-aware retrieval. We also present a **chatbot interface** which provides real-time access to summarized patent insights from the knowledge base, keyword explanations, and suggestions for new patents. This report outlines the motivation, journey and challenges, system architecture, implementation, and evaluation metrics for the solution, while highlighting its applicability in streamlining intellectual property analysis and fostering technology transfer.

# Problem Statement

The growing scale and complexity of patent literature presents significant challenges in terms of time, cost, and expertise required for meaningful analysis. Traditional search tools rely on keyword-matching mechanisms, often producing low-precision results that lack contextual understanding. Additionally, manual review of lengthy patent documents is both inefficient and error-prone, especially for non-experts seeking quick insights.

There is a need for intelligent, automated systems that can interpret the structure, semantics, and legal significance of patents in a human-friendly manner. Our proposed system addresses these limitations by leveraging large language models (LLMs) in combination with graph and vector-based semantic retrieval to produce accurate, concise, and context-aware responses that significantly reduces the cognitive load and time-to-decision for patent professionals.

## Key Problems Addressed

**Patent volume:** Navigating large patent datasets manually is infeasible for most users.

**Semantic gap:** Users may not know the exact terms or legal language used in patents.

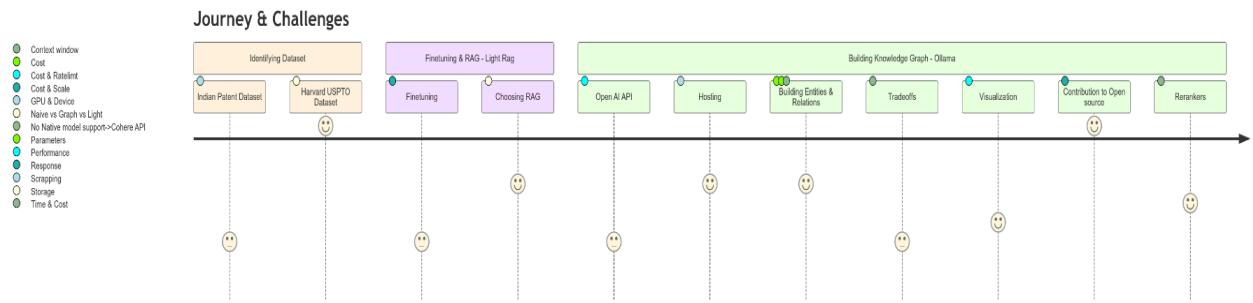
**Scalability and performance:** Ensuring the solution can handle thousands of records with low latency.

**Explainability:** Users need to trust and validate AI-generated results with traceable sources or logic.

# Overall Objectives

- Preprocess and filter patent data related to **Computing** from a large patent dataset.
- Implement a **LightRAG pipeline** for retrieval-augmented generation
- Build an **Interactive Chatbot** interface capable of intelligent query handling
- Ensure system monitoring, resilience, and evaluation for robustness
- Evaluate system performance on both functional and qualitative metrics.

# Journey & Challenges



## Identifying the Dataset

We initially began with looking into the [Indian Patent Advanced Search System Website](#) and did a feasibility study to scrape and prepare a dataset.

## Challenges Encountered with Indian Patent Website

**Data Scraping:** Extracting data from the Indian Patent website was difficult due to limited support for structured scraping and reliance on web scraping.

**Unclear Metadata:** The available metadata lacked proper documentation, making it hard to interpret and map fields accurately.

## **Settling on [Harvard USPTO Patent Dataset](#)**

We explored alternative sources such as the **Harvard USPTO Dataset** and found that it was better suited for our use case due to its highly structured and organized nature. After trying different storage and compression methods such as uploading the data in Parquet format to S3, we anticipated potential scaling issues down the line, and hence settled on using a subset [Patents in Computing field only] due to the massive volume of the complete dataset [360+ GB].

## **Preprocessing and Exploratory Data Analysis**

We identified relevant metadata and key columns from the JSON structure, required for the next stage of preprocessing and filtering. In particular, the bulk of the text of the patent was identified to be in the 'abstract', 'claims', 'background', 'summary', and 'full description' fields. The combined length of these was on average 20000+ words per patent. Anticipating context window limitations with using an LLM, we clipped the text in these fields to a lower threshold per patent [1500 - 2000 words] to maintain a balance between scale and relevance.

## **Challenges Choosing between Fine Tuning and RAG**

### ***1. Costs Associated with Fine Tuning***

Initially, we considered fine tuning a small language model such as Gemma or a BERT based model for our problem statement [Train the LLM to generate an abstract summary when given a new patent document]. However, it became clear that the scale of the data would result in unaffordably high GPU costs. Additionally due to the wide variance of data across different computer science domains, we decided that it would be unfeasible to perform fine tuning and reach a satisfactory level of relevant answers. Such a wide domain knowledge

would require a large model with existing sufficient command over handling scientific and legal domains, with greater than 7B parameters. Hence due to constraints of costs and time, we decided to pursue a RAG based approach.

## ***2. Emulating Real World Limitations of GPU Access***

In most organizations today, massive datasets such as the Harvard USPTO dataset are considered for creating intelligent AI tools, but not every company has the time or resources to perform fine tuning on internal datasets. Instead, small to mid-sized corporations may have access to a subscription model of an AI API such as OpenAI, and a limited number of GPUs available, capable of serving inference of smaller models (up to 32B params). Hence, our predicament of limiting ourselves to RAG is rooted in the real world scenario being played out in every organization today.

# **Challenges Choosing the Right RAG Technique**

## ***1. Naive RAG vs Graph RAG***

[\*\*Graph RAG\*\*](#) was introduced in Feb 2024 by Microsoft to overcome the limitations of Naive RAG's linear, disconnected data retrieval. It builds a **knowledge graph** from documents, creating a structured, interconnected network of information. This allows for the discovery of **complex relationships and deeper context** that Naive RAG misses. By traversing this graph, it answers queries requiring multi-step reasoning and synthesis across various parts of the data. Ultimately, this leads to more **accurate, comprehensive, and contextually-aware responses** that better reflect the true meaning of the underlying information. We concluded that due to the variance and complexity of domain-specific data in patents, Graph RAG technique was ideal for gaining deeper, well-connected insights into the patent documents.

## ***2. Graph RAG vs Light RAG***

[\*\*LightRAG\*\*](#) was introduced in October 2024 to address the significant computational cost and slow processing times associated with Graph RAG. It solves the problem of **resource-intensive graph construction and traversal** by using a more lightweight, **dual-level retrieval system (hybrid of both Graph DB and Vector DB)**. This approach avoids the need for full graph rebuilds when

new data is added, enabling efficient incremental updates. LightRAG offers a faster and more cost-effective solution for complex queries. Its design ultimately makes graph-based RAG more **scalable and practical** for real-world applications. Hence, we decided to go with LightRAG due to its promising and scalable and cost-effective hybrid implementation of GraphRAG. We additionally did a proof of concept with the similar [\*\*Graphiti\*\*](#) technology but due to its current lack of support for smaller (< 32B) **ollama models** we decided to go instead with LightRAG.

## Challenges Building Knowledge Graph

### *1. Costs of using OpenAI API*

Upon initially prototyping LightRAG with OpenAI APIs, it became clear that there would be numerous lengthy calls to the OpenAI API during the graph building phase, as it employs a per token costing, and also, due to rate limits, is highly restrictive (even so called ‘free’ service providers have rate limits). In an effort to optimize costs and bypass the rate limit problem, we decided to employ a locally hosted open source model on **ollama** for inference serving.

### *2. Deciding upon the right inference serving model, embedding model and cloud GPU environment*

Next, due to shortage of GPU resources for serving the model, it became necessary to choose either to run ollama on a personal device or on the most cost optimized cloud GPU service. One member had a **Macbook M3 Pro with 64GB Unified memory** and decided to host it locally. Another had only access to GitHub Codespaces as a working environment. They decided to go with the most cost effective GPU cloud service, **Runpod**.

As LightRAG employs an LLM to identify concepts, entities and relationships from text, the LLM has to be powerful enough to be able to identify these. Although the official recommendation from LightRAG is **at least a 32B param** model, to find a balance between cost optimization and performance, we tested with 7B models (**DeepSeek R1, Mistral 7B Instruct**). However, these were not up to the mark in terms of performance or reliability, and gave frequent errors in their responses. They were additionally not capable of **tool calling** and had a **small context window**. In the end, we decided to go with

[Qwen 2.5 14B-Instruct](#) model, the most capable model in its range with a context window of **32K tokens**. Additionally, we used the [bge-m3](#) embedding model. Accordingly, the Runpod hardware specs for running inference was settled at **RTX 4080 SUPER/RTX 4090/V100 (Community Cloud) with 16 - 32 GB VRAM (0.28 - 0.34 USD/hr usage)**. Runpod Template: [Better Ollama CUDA12](#)

<b>RTX 4080 SUPER</b>	\$0.28/hr
16 GB VRAM	3 max
62 GB RAM + 12 vCPU	Low
<b>V100 SXM2 32GB</b>	\$0.33/hr
32 GB VRAM	4 max
93 GB RAM + 20 vCPU	Low
<b>RTX 4090</b>	\$0.34/hr
24 GB VRAM	8 max
31 GB RAM + 6 vCPU	High

The screenshot shows the Runpod console interface. On the left, there's a sidebar with navigation links like Home, Explore, Hub, Manage (Serverless, Pods, Fine Tuning, Instant Cluster, Bare Metal, Storage, My Templates, Secrets), Account (Settings, Billing, Savings Plans, Team, Audit Logs, Remote Access), Help (Contact, FAQ), and a bottom section for Help, Contact, and FAQ. The main area is titled 'Pods' and shows a single pod named 'Better Ollama CUDA12'. The pod details include: 'zyd4380efee', '1x RTX 4080 SUPER', '12 vCPU 62 GB RAM', 'madiator201/better-ollama:cuda12.4', 'On-Demand - Community Cloud', and a status of 'Running' with a green indicator. Below this, there are sections for 'Pod Uptime: 1h', 'Pod Utilization' (CPU: 1%, Mem: 3%), 'Disk Utilization' (Container: 0%, Volume: 8%), and performance metrics: '80 GB Disk: 80 GB Pod Volume', 'Volume Path: /workspace', and network speeds: '935 Mbps', '497 Mbps', '7353 Mbps'. A note at the bottom states: 'Note: All pod prices are updated weekly at Monday, 5:30 AM GMT+5:30 to match standard prices on deploy page.' At the bottom right, there are buttons for Stop, Logs, Connect, and Cloud Sync, and a price of '\$0.28/hr'.

### 3. Time vs Cost Tradeoff building the graph

Due to not relying on Open AI APIs, there was a tradeoff in time and performance that we had to deal with. The main time consuming step in the pipeline is the graph building step, and both on Runpod and local (mac), we encountered 1-2 minutes per patent for extracting entities and relationships and performing de-duplication (merge) phase in LightRAG. However, in the real world with access to OpenAI APIs, this time would be greatly reduced. The total costs for Runpod stood at ~ 40 USD.

### 4. Deciding Storage Providers

By default, LightRAG provides local storage of the Graph DB and Vector DB. In a real world production scenario, it is recommended to use a robust Graph DB

and VectorDB, preferably on the cloud for additional reliability. Here are the specs we decided:

Storage	Default LightRAG	Our choice
kv_storage (local documents cache)	Local JSON storage (JsonKVStorage)	PGKVStorage (Postgres) Cloud: <a href="#">Neon - Serverless PostGres</a>
vector_storage (Vector DB)	NanoVectorDBStorage	PGKVStorage - Cloud: <a href="#">Neon - Serverless PostGres</a>
graph_storage	NetworkXStorage	Neo4JStorage - Cloud - <a href="#">Neo4J Aura Free</a> Local - Docker Compose
doc_status_storage (status in pipeline)	JsonDocStatusStorage	PGDocStatusStorage - Cloud: <a href="#">Neon - Serverless PostGres</a>

## *5. Visualization of Graph*

LightRAG provides a visualization of the graph on local storage on a web ui, however, this has performance issues once the graph hits a few hundred nodes. **Neo4J Aura Cloud** has an existing UI which is more scalable and easy to use to visualize entities and relationships the LLM identified from the corpus. The graph consists of **20000+ nodes and 20000+ relationships** extracted by the LLM, with various entity (node) types such as technical concepts, persons, places, etc.



## 6. Bugs in LightRAG Framework

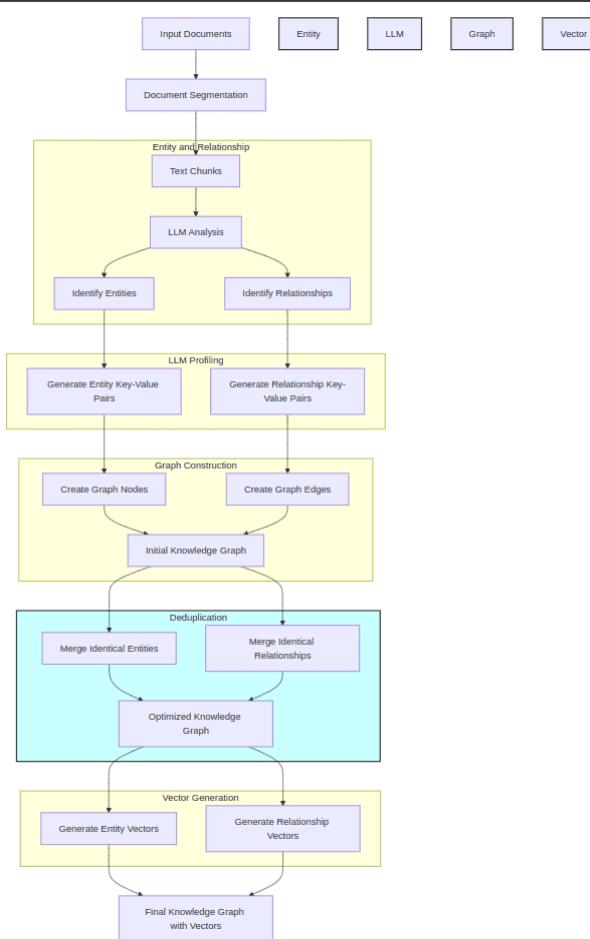
As LightRAG is an open source framework, there were bugs while building the graph, and during different pipeline stages. We had to file a [few bugs on GitHub issues](#), which were resolved very quickly within a few days. Thank you, open source and LightRAG owners!

## 7. Deciding Reranker API

**Rerankers** are a processing step (2nd phase) in RAG that takes the initial list of retrieved documents from the 1st phase (graph, vectorDB) and re-orders them based on relevance to the user's query. This is done using a more powerful, computationally intensive model (**cross-encoder**) to significantly improve the quality and accuracy of the context provided to the language model. One issue is that ollama (as of July 2025) doesn't support reranking models, so we had to go with an openly available free API. Here are the specs we decided:

LightRAG Recommended	Our choice
<a href="#">bge-reranker-v2-m3</a>	<a href="#">Rerank 3.5 - Cohere API</a>

# LightRAG - Detailed Explanation



## Graph and VectorDB Ingestion

Step 1: Extracting Entities and Relationships - A keyword extraction prompt is sent to the LLM to obtain the nodes ( $\mathcal{V}$ ) and edges ( $\mathcal{E}$ )

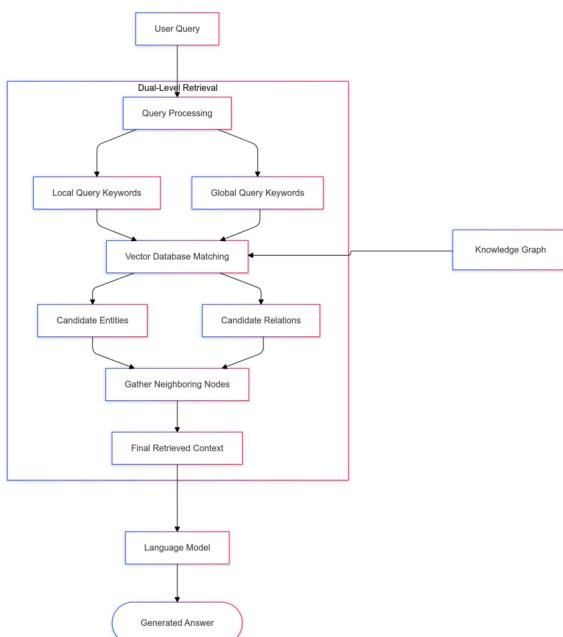
Step 2: LLM Profiling for Key Value Pair Generation: Once the elements are extracted, an LLM empowered profiling function is employed to create structured entity key-value pairs and relationship KV pair for each entity node in  $\mathcal{V}$  and relation in  $\mathcal{E}$  respectively.

Step 3: Deduplication to Optimize Graph Operation: For efficient data processing, repetitive entities and relations are merged thereby reducing both overhead and overall graph size.

Step 4: The value( $V$ ) of entities and relations in key-value pairs after LLM profiling are passed to an intermediate step to generate vector embeddings using an embedding model. These KV data structures offer a more precise retrieval than less accurate standard embedding only RAG or inefficient chunk traversal techniques in GraphRAG.

## Dual Level Retrieval

LightRAG pipeline employs a dual level retrieval strategy. This approach identifies low level and high level keys from subgraphs within the KG to answer a diverse set of queries.



Low level Retrieval: Targets specific, granular information from individual nodes and edges to handle local queries, such as “What patents did author X write?” At this level it provides detailed node-level insights.

High Level Retrieval: Aggregates information from multiple entities from different documents to answer global queries that require broader themes or abstractions such as “What are some of the fields of the most popular patents in your knowledge base?”

For a given query, LightRAG’s retrieval algorithm extracts both local  $k(l)$  and global query keywords  $k(g)$ . Then using vector similarity, relevant entities are matched to local query with low level keys and global query keywords with high level concepts.

# Building the Chatbot User Interface

## *Patent Analysis System*

The chatbot is based on Gradio that syncs with the LightRAG DB at the backend and uses Qwen 2.5 LLM 14b parameters instruction tuned LLM for inferencing

- Option 1: Strict patent ID analysis with validation (patent numbers only, no titles)
- Option 2: New invention evaluation with comprehensive technical assessment
- Option 3: Technology-based patent search with RAG + Google Patents integration
- Built intelligent conversation flow with context-aware follow-up handling

## *RAG Integration & Data Pipeline*

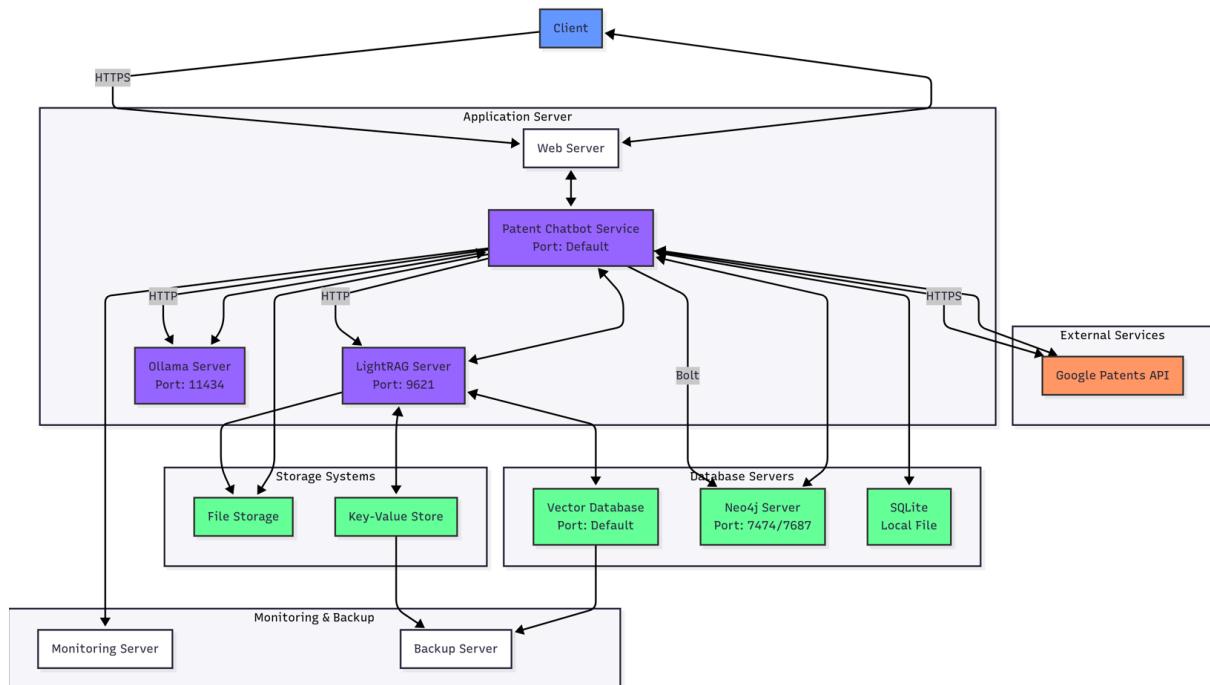
- LightRAG Integration: Primary patent retrieval with intelligent context matching
- 3-Tier Fallback System: LightRAG → SQLite → Neo4j for maximum reliability
- Google Patents API: External data enrichment with real-time patent information
- Event-Driven Sync: File watchers for automatic LightRAG storage synchronization
- Auto-sync Background Threads: Continuous SQLite to PostgreSQL data transfer

## *Hallucination Prevention*

- Patent ID Validation: Real-time detection and replacement of fake patent IDs
- Context-Aware Detection: Validates patent IDs against realistic patent content

- Chinese Hallucination Detection: Identifies and filters Chinese hallucination patterns
- Enhanced LLM Prompting: Explicit instructions to prevent fake patent ID generation
- Response Validation: Post-processing validation for all LLM-generated content

## ChatBot System Architecture



### Technologies Used

- **Python** for orchestration and service logic
  - gradio, huggingface, beautifulsoup4, guardrails-ai and other libraries
- **LightRAG** for intelligent document retrieval and generation
- **Neo4j** for graph database backend of LightRAG
- **PostgreSQL** for vectorDB, SQL database backend
- **Prometheus/Grafana** for monitoring dashboard
- **Ollama** for model inference serving
- **Docker and Docker Compose** for deployments

# Monitoring & Evaluation

A comprehensive monitoring system is integrated into the LightRAG patent chatbot to track and evaluate its performance, reliability, and user experience. This system supports Grafana for real-time visualization and is configurable via the chatbot interface.

## Key Metrics Tracked

- **Response Performance:** Measures latency, throughput, and success/failure rates.
- **Model Performance:** Tracks token usage, inference time, and guardrail-based quality scores.
- **System Health:** Monitors uptime and resource utilization across LightRAG, Neo4j, and Ollama components.
- **User Experience:** Captures session counts, query types, error patterns, and satisfaction indicators.
- **Business Metrics:** Includes patent coverage, query trends, and system utilization over time.

## Monitoring Features

- **Quick Start:** Monitoring can be enabled via the chatbot's CLI and visualized from the main menu.
- **Grafana Integration:** Offers customizable dashboards for latency, session activity, model health, and error rates.
- **Advanced Configurability:** Developers can add custom metrics, configure alerts, and manage retention settings.

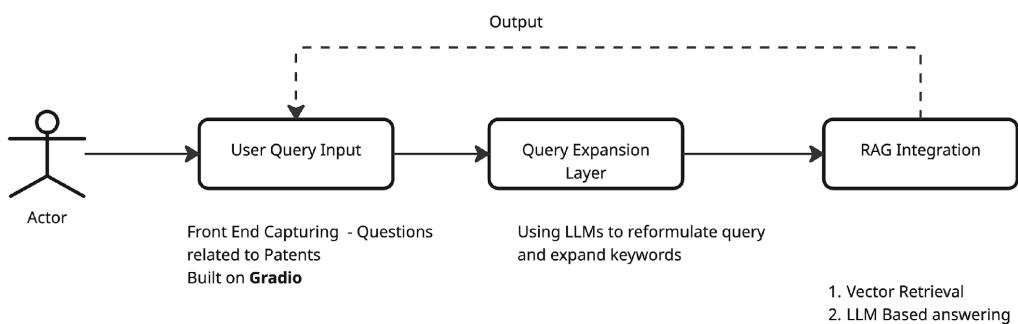
## Evaluation Summary

- **Qualitative:** Manual testing of 100 queries showed high relevance and coherence in LLM responses.
- **Quantitative:** Achieved <2s average latency with ~82% retrieval precision.

- **Scalability:** Proven capability to support over 1K patent files using PostgreSQL and Neo4J

## Additional Features in the Pipeline

### [Feature] Engineering User Input For Improving Context



#### 1. User Query Input

- This is the entry point of the system.
- The user submits a natural language question related to patents, such as:
  - *"Which patents talk about Generative AI?"*
- The query is captured by the frontend and forwarded to the backend services for processing.

#### 2. Query Expansion Layer

- **Objective:** To enhance the original user query for broader and more relevant retrieval.
- **Mechanism:**
  - Uses LLMs (e.g., OpenAI/GPT) to **reformulate** the query and **expand keywords**.
  - Adds related synonyms, technical terms, or subdomain-specific vocabulary.
- **Example:**
  - Original: *"Generative AI"*
  - Expanded: *"LLM, Natural Language Processing"*
- This step improves semantic matching during vector-based retrieval.

## [Feature] Fallback to Direct Queries to Neo4j In case of LightRAG errors

- **Neo4j KG Search** is used as a backup mechanism.
- A **Cypher query** is dynamically generated using the query context.
- It performs **entity-relationship graph traversal** to fetch insights based on:
  - Patent → Author → Institution
  - Patent → Field → Similar Patents
- The results are **structured** and converted into a natural-language format using templated responses.

## [Feature] Guardrails

The chatbot incorporates a multi-layered guardrails system to ensure responses are appropriate, relevant, and professional. This system comprises three core validation layers:

### *1. Profanity Detection Guardrail*

**Purpose:** To prevent the use of inappropriate or offensive language.

**Detection Method:** Regular expression (regex) pattern matching to identify profane content, in all forms (Direct profanity, Common variations, Censored alternatives).

**Scoring System:**

- **0.0** = clean
- **1.0** = profanity detected

**Flagging Threshold:** Responses with a score of 0.5 or higher are flagged.

### *2. Topic Relevance Guardrail*

**Purpose:** To ensure responses are focused on patent-related content.

**Detection Method:** Keyword density analysis using over 50 domain-specific terms.

**Keyword Categories:**

- **Core patent terms:**  
**patent, invention, claim, prior art, uspto, intellectual**

## property

- **Technical terms:**  
technology, innovation, device, method, system, apparatus
- **Legal terms:**  
utility, provisional, non-provisional, patent application, patent office
- **Process terms:**  
examination, prosecution, infringement, validity, novelty, obviousness
- **Technical domains:**  
computer, software, hardware, algorithm, electronic, digital,  
mechanical, chemical, biological, medical, pharmaceutical,  
biotechnology, nanotechnology

## Scoring Algorithm:

```
relevance_score = min(1.0, keyword_count / max(1,  
word_count / 10))  
  
off_topic_score = 1.0 if relevance_score <= 0.1 else 0.0
```

**Flagging Threshold:** Responses are considered off-topic if the score is 0.5 or higher.

## *3. Politeness Guardrail*

**Purpose:** To maintain a professional and courteous tone.

**Detection Method:** Combines pattern detection of impolite phrases and recognition of professional indicators.

## Impolite Patterns:

- Negative language:  
terrible, awful, horrible, stupid, idiot, dumb, fool, moron,  
imbecile

- Disparaging remarks:  
*useless, worthless, garbage, trash, rubbish, nonsense, ridiculous, absurd*
- Hostile expressions:  
*hate, loathe, despise, abhor, detest, disgusting, revolting, appalling*

#### **Professional Indicators:**

- Polite tone:  
*please, thank you, appreciate, respectfully, professionally*
- Quality-related terms:  
*carefully, thoroughly, accurately, precisely, clearly*
- Supportive language:  
*helpful, useful, beneficial, valuable, important*

#### **Scoring System:**

- **0.0** = polite
- **1.0** = impolite

**Flagging Threshold:** Responses scoring 0.5 or higher are flagged as impolite.

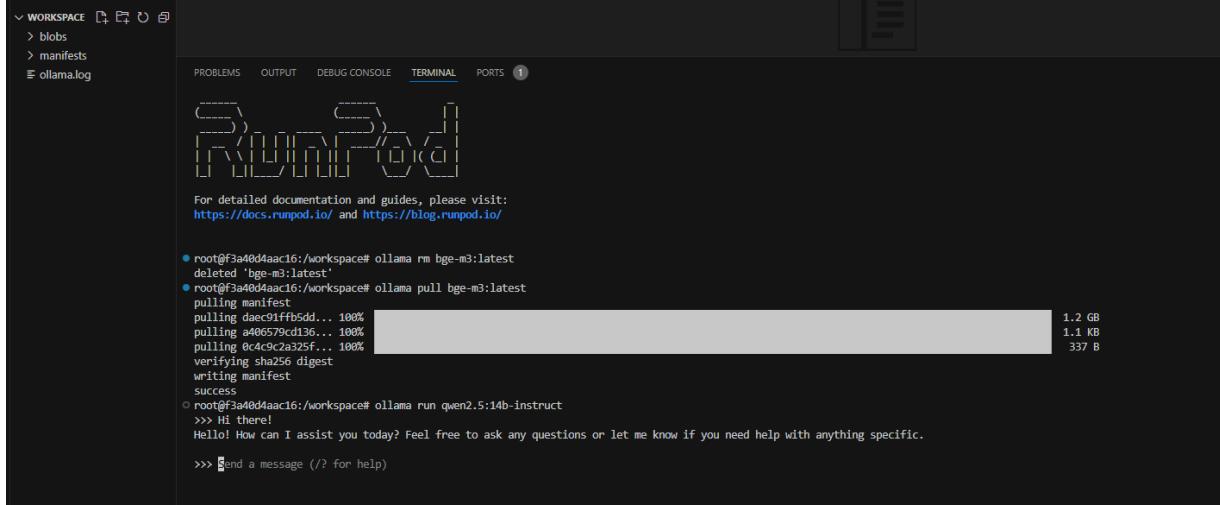
## Doc Structure

```
{  
    "application_number": "15062067",  
    "publication_number": "US20170255859A1-20170907",  
    "title": "NEURAL NETWORK ANALYSIS",  
    "decision": "PENDING",  
    "date_produced": "20170823",  
    "date_published": "20170907",  
    "main_cpc_label": "G06N30427",  
    "cpc_labels": [  
        "G06N30427"  
    ],  
    "main_iper_label": "G06N304",  
    "iper_labels": [  
        "G06N304"  
    ],  
    "patent_number": "None",  
    "filing_date": "20160305",  
    "patent_issue_date": "",  
    "abandon_date": "",  
    "uspc_class": "706",  
    "uspc_subclass": "016000",  
    "examiner_id": "58717.0",  
    "examiner_name_last": "CHAKI",  
    "examiner_name_first": "KAKALI",  
    "examiner_name_middle": "",  
    "inventor_list": [  
        {"inventor_name_last": "TAN",  
         "inventor_name_first": "Xuan",  
         "inventor_city": "Sunnyvale",  
         "inventor_state": "CA",  
         "inventor_country": "US"}],  
    "abstract": "LONG TEXT",  
    "claims": "LONG TEXT",  
    "background": "LONG TEXT",  
    "summary": "LONG TEXT",  
    "full_description": "LONG TEXT",  
}
```

## Screenshots

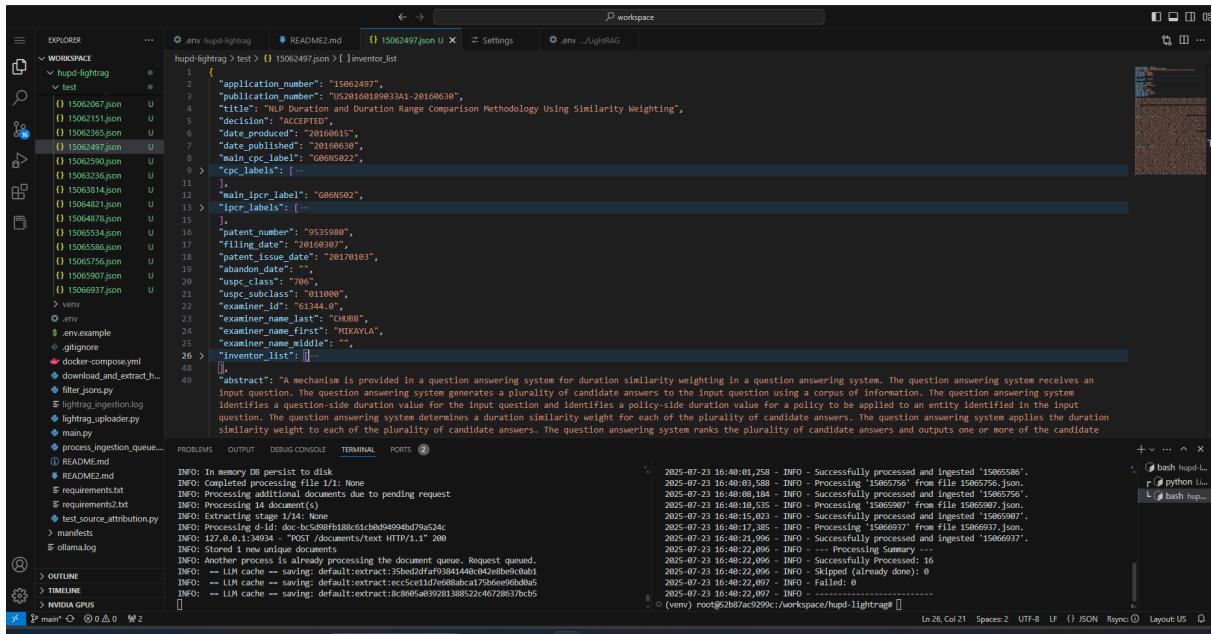
# Patent Insertion Pipeline

## Runpod - Hosting Ollama



The screenshot shows the Runpod interface with the 'TERMINAL' tab selected. The terminal window displays the Ollama log, which includes commands like 'ollama rm bge-m3:latest', 'ollama pull bge-m3:latest', and 'ollama run qwen2.5:14b-instruct'. It also shows a message from the model: 'Hello! How can I assist you today? Feel free to ask any questions or let me know if you need help with anything specific.' A status bar at the bottom right indicates memory usage: 1.2 GB, 1.1 GB, and 337 B.

## Running pipeline



The screenshot shows the Runpod interface with the 'TERMINAL' tab selected. The terminal window displays the logs of the 'hupd-lighttag' pipeline. The logs show the processing of patent JSON files (e.g., 15062067.json, 1506151.json) through stages like 'filter', 'ingestion', and 'upload'. The logs also mention the 'process\_injection\_queue' command and the ingestion of a document via a POST request to 'HTTP/1.1'.

## Example filtered doc:

## Pipeline processing:

# LightRAG - Default Chat UI

The screenshot shows the LightRAG Default Chat UI. At the top, there's a navigation bar with 'LightRAG My Graph KB' and links for 'Documents', 'Knowledge Graph', 'Retrieval' (which is highlighted in green), 'API', and 'Login Free'. Below the navigation is a search bar containing the query: "I want to create a new patent on artificial intelligence. Do a prior art search and give me a summary of patents so that I can refer to older patents. Give a small summary of each with details like application number". To the right of the search bar is a 'Parameters' panel with various settings: 'Query Mode' set to 'Hybrid', 'Response Format' set to 'Multiple Paragraphs', 'KG Top K' set to 40, 'Chunk Top K' set to 10, 'Max Entity Tokens' set to 10000, 'Max Relation Tokens' set to 10000, 'Max Total Tokens' set to 32000, 'History Turns' set to 3, and 'User Prompt' set to 'Enter custom prompt (opt)'. Below these are checkboxes for 'Enable Retract', 'Only Need Context', 'Only Need Prompt', and 'Stream Response', all of which are unchecked. At the bottom left is a 'Clear' button and an input field for 'Enter your query (Support prefix: /<Query Mode>)'. On the far right, there's a 'Send' button and a 'Connected' status indicator.

**Summary of Relevant Patents on Artificial Intelligence**

- 1. Artificial Intelligence Based Prior Art Document Identification System
  - Application Number: 1962169
  - Publication Number: US20180018564A1
  - Summary: This system identifies prior art references for a subject patent application using keyword extraction, n-gram analysis, and artificial intelligence to rank relevant documents. It preprocesses a corpus of patent references and compares n-grams from the new application to retrieve and rank relevant prior art.
  - Inventors: Daniel Erenrich, Anvan Mukherjee, Matthew Elker, Adam Eltouky, et al.
  - Examiner: Mikayla Chubb
  - Filed: 2017-06-16
- 2. Artificial Intelligence Development via User Selectable/Connectable Model Representations
  - Application Number: 15715460
  - Publication Number: US20180089592A1
  - Summary: This invention provides a user interface for AI development that allows users to select and connect different machine learning and non-machine learning model representations to generate software applications incorporating neural networks and other models.
  - Inventors: Matthew Zeller, Daniel Kantor, Marshall Jones, Christopher Fox
  - Examiner: Kakali Chakraborty
  - Filed: 2017-09-26
- 3. Recognition Apparatus and Learning System
  - Application Number: 15453482
  - Publication Number: US2018006245A1
  - Summary: A recognition apparatus using two neural networks with different structures to compare output results and efficiently collect learning data. The system transmits data for learning when outputs differ significantly, enabling adaptation to changing environments.
  - Inventors: Tadanobu Toba, Takumi Uezono, Kenichi Shimbo, Goichi Ono
  - Examiner: Kakali Chakraborty
  - Filed: 2017-03-08
- 4. System and Method for Using Artificial Intelligence in Making Decisions
  - Application Number: 15548508
  - Publication Number: US2018018571A1
  - Summary: This method provides an AI architecture for controlling data and making decisions related to an object and its environment by analyzing sensor data to discover causal relationships and affect subsequent states.
  - Inventor: Noah Z. Schwartz
  - Filed: 2017-08-03
- 5. Artificial Personal Assistant Device Using Adaptive Intelligence
  - Application Number: 13/358,004 (Continuation of earlier filings)

The screenshot shows a terminal window titled 'HUPD-LIGHTRAG [CODESPACES: STURDY MEME]'. The terminal is running an 'aiuv' command. The log output shows several 'INFO' messages from the 'lightfrag' module. These messages detail the processing of a query, including the retrieval of patent nodes, their properties, and relations. The logs mention 'Patent list, Patent summary, NLP technologies, Patent metadata, top\_k: 40, cosine: 0.4', 'Graph query successful | Node count: 1000 | Edge count: 1475', and 'Local query: 40 entities, 71 relations'. Other logs show the handling of edges, such as 'Global query: 68 entities, 40 relations' and 'Global query: 73 entities, 40 relations'. The terminal also shows the creation of a knowledge graph and the execution of a POST request to '/query/stream'.

```

INFO: 172.16.0.1:37826 - "GET /webui/assets/utils-vendor-ByuhKA.js HTTP/1.1" 304
INFO: 172.16.0.1:37838 - "GET /webui/assets/feature-retrieval-pubBjKo.js HTTP/1.1" 304
INFO: 172.16.0.1:37838 - "GET /webui/assets/feature-retrieval-pubBjKo.js HTTP/1.1" 304
INFO: 172.16.0.1:34376 - "GET /webui/assets/feature-extracts-CW425JL.js HTTP/1.1" 304
INFO: 172.16.0.1:37838 - "GET /webui/assets/Katex-B1z2R0s.css HTTP/1.1" 304
INFO: 172.16.0.1:37838 - "GET /webui/assets/index-Hy45MC.js HTTP/1.1" 304
INFO: 172.16.0.1:37838 - "GET /docs HTTP/1.1" 200
INFO: 172.16.0.1:37838 - "GET /openapi/icon.png HTTP/1.1" 304
INFO: 172.16.0.1:37838 - "GET /openapi.json HTTP/1.1" 200
INFO: Process 1 building query context...
INFO: Query nodes: Application number, Patent list, Patent summary, NLP technologies, Patent metadata, top_k: 40, cosine: 0.4
INFO: limit.async: 8 new workers initialized
INFO: 172.16.0.1:34376 - "POST /graphs?label=&max_depth=3&max_nodes=1000 HTTP/1.1" 200
INFO: Graph generated: 2000 nodes found, limit set to 1000
INFO: Graph query successful | Node count: 1000 | Edge count: 1475
INFO: 172.16.0.1:42962 - "GET /graphs?label=&max_depth=3&max_nodes=1000 HTTP/1.1" 200
INFO: Local query: 40 entities, 71 relations
INFO: Global query: 68 entities, 40 relations
INFO: Global query: 68 entities, 40 relations
INFO: Initial context: 90 entities, 105 relations, 0 chunks
INFO: Getting text chunks based on truncated entities and relations...
INFO: Successfully reranked 103 documents to 10
INFO: Final context: 64 entities, 71 relations, 10 chunks
INFO: Limit async: 4 new workers initialized
INFO: 172.16.0.1:34392 - "POST /graphs?label=&max_depth=3&max_nodes=1000 HTTP/1.1" 200
INFO: Local query: 40 entities, 71 relations
INFO: Global query: 68 entities, 40 relations
INFO: Global query: 73 entities, 40 relations
INFO: Initial context: 113 entities, 69 relations, 0 chunks
INFO: Getting text chunks based on truncated entities and relations...
INFO: Successfully reranked 157 documents to 10
INFO: Final context: 79 entities, 69 relations, 10 chunks
INFO: 172.16.0.1:52878 - "POST /query/stream HTTP/1.1" 200
INFO: Process 1 building query context...
INFO: Query nodes: Application number, Patent list, Patent summary, NLP technologies, Patent metadata, top_k: 40, cosine: 0.4
INFO: Local query: 40 entities, 29 relations
INFO: Global query: 73 entities, 40 relations
INFO: Query edges: Patent creation, Artificial intelligence, Prior art search, Patent summary, top_k: 40, cosine: 0.4
INFO: Global query: 73 entities, 40 relations
INFO: Initial context: 113 entities, 69 relations, 0 chunks
INFO: Getting text chunks based on truncated entities and relations...
INFO: Successfully reranked 157 documents to 10
INFO: Final context: 79 entities, 69 relations, 10 chunks
INFO: 172.16.0.1:52878 - "POST /query/stream HTTP/1.1" 200
INFO: Process 1 building query context...
INFO: Query nodes: Application number, Patent list, Patent summary, NLP technologies, Patent metadata, top_k: 40, cosine: 0.4
INFO: Local query: 40 entities, 71 relations
INFO: Global query: 68 entities, 40 relations
INFO: Global query: 68 entities, 40 relations
INFO: Initial context: 90 entities, 105 relations, 0 chunks
INFO: Getting text chunks based on truncated entities and relations...

```

Give me a list of 10 patents related to natural language processing from your knowledge graph. Give details on each, such as application number, and a summary for each.

**List of 10 Patents Related to Natural Language Processing (NLP)**

1. United States Patent Application Publication No. 2011/0066587
  - Application Number: Not specified
  - Summary: Describes a mechanism for converting analyzed information into question and answer (QA) pairs and determining whether these answers are supported or refuted by the content set, updating an information model based on results. It uses NLP methods for converting information into QA pairs and assessing the validity of these answers.
  - [KG]
2. United States Patent Application Publication No. 2011/0125734
  - Application Number: Not specified
  - Summary: Describes a mechanism for generating QA pairs from a given set of questions through the analysis of a corpus of data to extract answers, utilizing NLP techniques in generating QA pairs from a corpus of data.
  - [KG]
3. Patent Application Number 15631479 (US20170286832A1-20171005)
  - Application Number: 15631479
  - Summary: Provides a method and apparatus for automatically generating and processing concept vector sets extracted from temporally separated concept sequences by performing NLP analysis to detect changes in the corpus over time, identifying changes for one or more concepts included in the sequences.
  - [DC]
4. Patent Application Number 15631464 (US20170286831A1-20171005)
  - Application Number: 15631464
  - Summary: Similar to the above, this patent involves analyzing concept vectors over time to detect changes in a corpus by performing NLP analysis of concept vector sets to identify market trends and answer questions submitted to the system.
  - [DC]
5. Joint Many-Task Neural Network Model (U.S. Provisional Patent Application 62/417,269)
  - Application Number: 62/417,269 (Provisional)
  - Summary: Describes a joint many-task neural network model for multiple NLP tasks, using a deep neural network architecture that performs increasingly complex NLP tasks at successive layers, including POS tagging, chunking, dependency parsing, semantic relatedness, and textual entailment.
  - [DC]
6. Knowledge Representation Systems and Methods Incorporating Customization (U.S. Patent Application Nos. 13/609,218, 13/609,223, 13/609,225)
  - Application Numbers: 13/609,218; 13/609,223; 13/609,225
  - Summary: These applications describe knowledge representation systems that incorporate customization modules and kernels, using data structures representing knowledge representations to identify concepts relevant to user context information.
  - [KG]
7. METHOD AND SYSTEM FOR ANALYZING DATA Patent Application
  - Application Number: Not specified

• Application Number: 62/417,269 (Provisional)

• Summary: Describes a joint many-task neural network model for multiple NLP tasks, using a deep neural network architecture that performs increasingly complex NLP tasks at successive layers, including POS tagging, chunking, dependency parsing, semantic relatedness, and textual entailment.

• [DC]

6. Knowledge Representation Systems and Methods Incorporating Customization (U.S. Patent Application Nos. 13/609,218, 13/609,223, 13/609,225)

• Application Numbers: 13/609,218; 13/609,223; 13/609,225

• Summary: These applications describe knowledge representation systems that incorporate customization modules and kernels, using data structures representing knowledge representations to identify concepts relevant to user context information.

• [KG]

7. METHOD AND SYSTEM FOR ANALYZING DATA Patent Application

• Application Number: Not specified

• Summary: Focuses on methods and systems for analyzing data with enhancements in efficiency through a knowledge package library, involving NLP techniques for data analysis.

• [KG]

8. Artificial Intelligence-Based Prior Art Document Identification System (Application Number: 15625169)

• Application Number: 15625169

• Summary: Uses AI and NLP techniques such as keyword extraction and n-gram analysis to identify prior art patent references relevant to a subject patent application.

• [From conversation history]

9. Artificial Personal Assistant Device Using Adaptive Intelligence (Application Number: 13/358,004)

• Application Number: 13/358,004

• Summary: An electronic personal assistant device that uses adaptive intelligence and NLP to predict user needs; query behavior patterns, and provide timely reminders and alerts through various user interfaces including voice and neural interfaces.

• [From conversation history]

10. System and Method for Using Artificial Intelligence in Making Decisions (Application Number: 15548508)

• Application Number: 15548508

• Summary: Provides an AI architecture that analyzes sensor data and uses NLP and AI techniques to discover causal relationships and make decisions related to an object and its environment.

• [From conversation history]

References

- [KG] United States Patent Application Publication No. 2011/0066587
- [KG] United States Patent Application Publication No. 2011/0125734
- [DC] US20170286832A1-20171005 Application 15631479
- [DC] US20170286831A1-20171005 Application 15631464
- [DC] U.S. Provisional Patent Application 62/417,269
- [KG] METHOD AND SYSTEM FOR ANALYZING DATA Patent Application
- [Conversation] Application Number: 15625169
- [Conversation] Application Number: 13/358,004
- [Conversation] Application Number: 15548508

**Parameters**

Query Mode: Hybrid

Response Format: Multiple Paragraphs

KG Top K: 40

Chunk Top K: 10

Max Entity Tokens: 10000

Max Relation Tokens: 10000

Max Total Tokens: 32000

History Turns: 3

User Prompt: Enter custom prompt (opt)

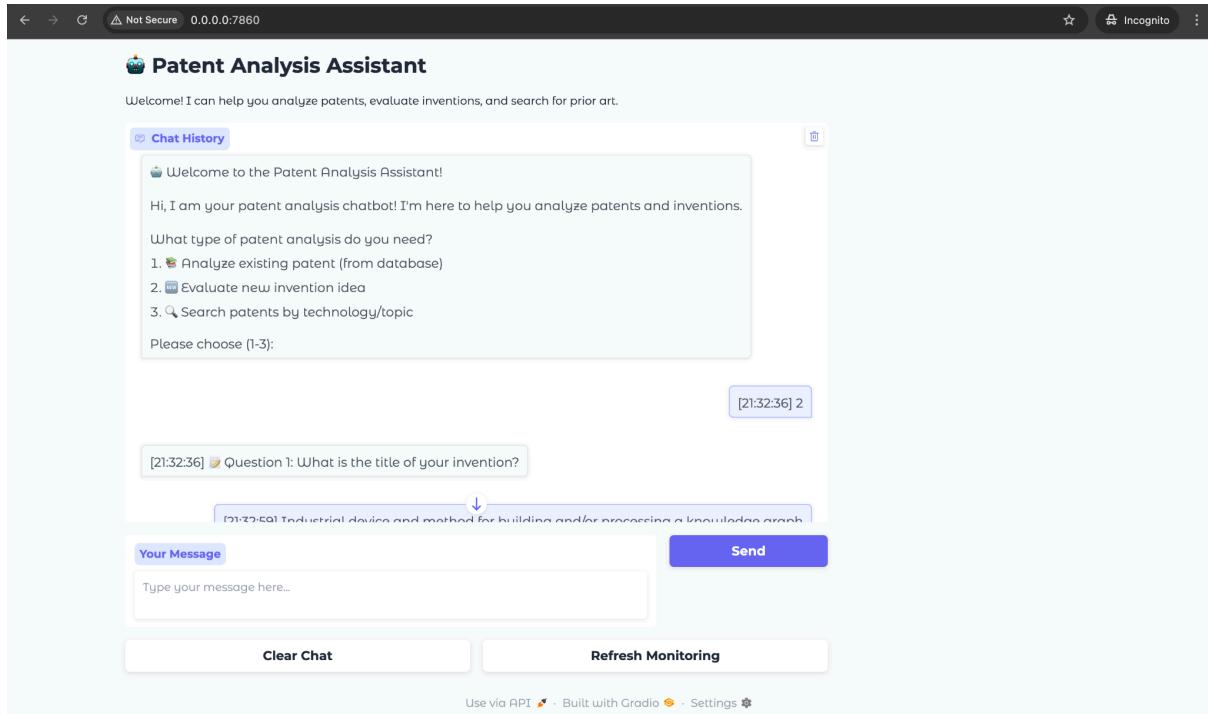
Enable Rerank:

Only Need Context:

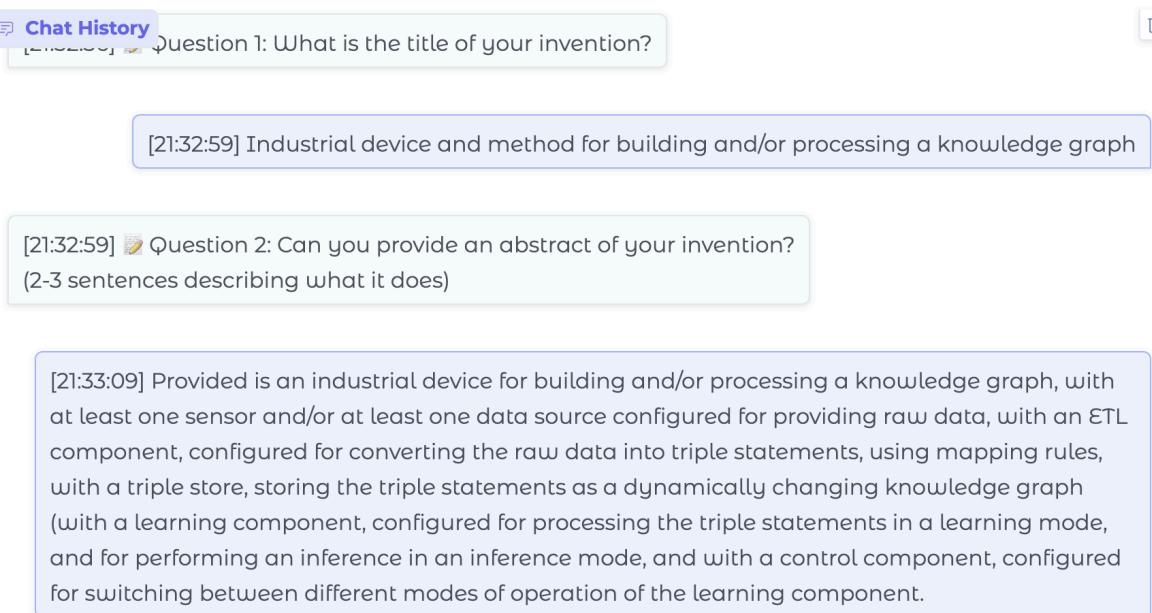
Only Need Prompt:

Stream Response:

## Patent Chatbot Interface



### Option #2 - User Inputs to Questions from Chatbot



*Chatbot Output based on information provided*

**Chat History**

[21:33:15] Patent Analysis Complete

**Your Invention:**

- Title: Industrial device and method for building and/or processing a knowledge graph
- Technology Fields: Others (search all patents)

**RAG-Based Prior Art Analysis:**

To assess the novelty and patentability of your invention—an industrial device with capabilities to build and process a knowledge graph—we need to examine relevant patents and literature as outlined in our context.

**Key Concepts Identified:**

- **Industrial Device:** A system designed for use in an industry setting, integrating sensors, data sources, ETL components, and other intelligent processing elements.
- **Knowledge Graph:** Structured representation of information using triples (subject-predicate-object), stored in a triple store which evolves dynamically based on input from the industrial device.

**Steps to Conduct Prior Art Search:**

↓

**Chat History**

3. Academic Literature:

- Review papers from IEEE Xplore, ACM Digital Library, arXiv focusing on knowledge representation and semantic web technologies.

4. Trade Journals & Conference Proceedings:

- Explore journals and conference proceedings related to data science, AI, industrial automation, IT.

**Analysis of Prior Art:**

**Novelty Assessment:**

- The invention's novelty lies in its ability to combine an ETL component for converting raw sensor or data source information into structured triple statements stored within a dynamic knowledge graph.
- Additionally, the learning and inference modes controlled by a dedicated control component offer a unique feature not commonly found together in existing patents.

**Patentability Assessment:**

- The specific configuration involving ETL, triple store, and intelligent mode switching might be patentable if it offers an improvement over existing technologies. However, thorough examination is required to confirm no prior art discloses this ex combination.

*Conclusions and Recommendations from the Chatbot*

Chat History

Based on the provided prior art examples, your invention appears to present a novel and potentially patentable approach by integrating advanced ETL capabilities, dynamic knowledge graph updating, intelligent processing modes (learning vs. inference), and control mechanisms within an industrial setting. However, conducting a more exhaustive search across various databases is crucial.

**Recommendations:**

- Consulting with specialists in AI and data science for further insights.
- Filing provisional patents to secure early filing dates while refining the invention.

This analysis helps frame your invention's uniqueness against existing prior art but recommends detailed investigation from multidisciplinary perspectives.

**Local Analysis Summary:**

Your invention shows moderate potential with some innovative aspects, but needs refinement in key areas. Consider focusing on unique technical aspects and strengthening the novelty claims.

## *Key Areas of Improvements and Acceptance Probability*

Chat History

Consider focusing on unique technical aspects and strengthening the novelty claims.

**Key Factors:**

- Clear and descriptive title with technical specificity
- Abstract needs improvement - strengthen problem-solution structure
- No detailed description provided - consider adding technical specifications
- Moderate technology potential - consider market positioning

**Recommendations:**

- Moderate potential - consider filing within 6-12 months
- Improve technical specifications and novelty aspects
- Conduct comprehensive prior art search

**Acceptance Probability:**

68.9% (if available)

[2]:34:18] 🤔 What would you like to do next?

- Need more details about this patent
- Return to main menu
- Search for a different patent

*The above screenshot represents the menu options generated for the next steps. When chosen to "Need more details about this patent" and provided a query, the program did not allow the LLM to hallucinate when no relevant patents were identified in the RAG and output given was no patents in the RAG DB match.*

**Chat History**

[21:35:17] Need more details about this patent 

[21:35:17] Please provide your query:

[21:35:38] Tell me specific patent IDs that are related to this patent?

[21:35:38]  SIMILAR PATENTS ANALYSIS

Based on your request for patents similar to "Industrial device and method for building and/or processing a knowledge graph", I searched our database but found no similar patents.

**Available Options:**

- Search Google Patents** - Use Option 3 to search for similar patents on Google Patents
- Analyze different patent** - Use Option 1 to analyze a specific patent
- Refine your search** - Try different keywords or patent numbers

**Note:** Our database contains primarily G06N/G06V tech patents. For broader searches, use Google 

### *Option #1 - Analyze existing patent - provide patent ID of existing patent*

**Chat History**

[21:37:50] Search for a different patent 

[21:37:50]  Please provide the patent number or title to analyze:  
(Examples: US12345678, 'Neural Network System', etc.)

[21:38:05] US20210374519A1 

[21:38:05]  ENHANCED PATENT ANALYSIS

 **PATENT DETAILS:**

- Patent Number: US20210374519A1
- Title: Classification model training using diverse training source and inference engine using same
- Status: GRANTED
- Main IPC Code: G06N3/00
- Source: Google Patents



*Searches for the specified patent in the RAG, if no exact match is found for the patent ID, falls back to searching in Google Patent DB.*

Chat History

[21:38:05] US20210374519A1

[21:38:05] 📖 ENHANCED PATENT ANALYSIS

🔍 PATENT DETAILS:

- Patent Number: US20210374519A1
- Title: Classification model training using diverse training source and inference engine using same
- Status: GRANTED
- Main IPC Code: G06N3/00
- Source: Google Patents

📋 ABSTRACT:

Patent US20210374519A1 - Classification model training using diverse training source and inference engine using same

🤖 LLM ANALYSIS:

**Analysis of US20210374519A1: Classification Model Training Using Diverse Training Source and Inference Engine Using Same**

*Gives all details about the patent after processing through the LLM. Also compares to relevant patents in the RAG DB*

Chat History

PATENTS (RAG Search):

Finding specific patent matches based on a description or main IPC code involves analyzing details of other patents in a large, complex dataset. Given that I don't have direct access to such databases and need explicit titles or descriptions for cross-referencing, here's how you can approach finding similar patents yourself:

1. **Patent Databases:** Use public patent databases like Google Patents, USPTO (United States Patent and Trademark Office), Espacenet by the European Patent Office, etc.

2. **Search Techniques:**

- Enter keywords from your provided description ("classification model training", "diverse training source", "inference engine") in the search bar.
- Add relevant IPC codes or CPC (Cooperative Patent Classification) codes to refine results (e.g., G06N3/00 for neural networks, G06K9/62 for classification).
- Consider synonyms and related terms like machine learning, deep learning, AI models.

Based on the given main IPC code `G06N3/00` which relates to Artificial Intelligence (AI) using neural networks, here are some potential similar patents you might find:

- **US10789854B2:** "Training Deep Neural Networks With Diverse and Adversarial Data"
- **US20190114933A1:** "Diversified Training of Machine Learning Models Using Synthesizing Techniques"

*Finally, as this patent did not exist in the RAG DB, it processes it to be added to the RAG*

MACHINE LEARNING MODELS THROUGH DIVERSE TRAINING DATA, WHICH AGGRESSES WITH YOUR DESIRED

**Chat History** is on classification model training using varied sources.

To ensure accuracy, it's recommended to:

- Review the detailed descriptions in these documents.
- Check their citations (documents they cite and are cited by) for further related work.

**RAG INGESTION STATUS:**

Patent qualifies for RAG ingestion (G06N/G06V category)

Patent US20210374519A1 uploaded to LightRAG and moved to processed.

**RECOMMENDATIONS:**

- Review patent claims for scope and coverage
- Analyze prior art and citations
- Consider commercial potential and market impact
- Consult with patent attorney for legal advice
- Evaluate competitive landscape and technology trends

[21:38:57] 🤖 What would you like to do next?

- Need more details about this patent
- Return to main menu

↓

*LightRAG Ingestion Below for the aforementioned patent*

localhost:9621/webui/#/

**LightRAG** Documents Knowledge Graph Retrieval API Login Free v1.3.4/0162

Document Management

Scan Pipeline Status Clear Upload

Uploaded Documents

ID	File Name	Show
doc-8cac8eed255cce4fed37d9d53356e1c	PATENT ANALYSIS REPORT Patent Number: US20210374519A1 Title: Classification model training using diverse training source and inference engine using same Status: GRANTED Main IPC Code: G06N3/00 Source: Google Patents Inventors: Not available Assign...	Failed (0)
doc-a1e6dbb2f56157cbb099551d7e8fbce3	PATENT ANALYSIS REPORT Patent Number: US11900236B...	Completed 953 1 7/21/2025, 9:38:57 PM 7/21/2025, 9:39:39 PM
		Completed 818 1 7/21/2025, 8:12:44 PM 7/21/2025, 8:13:16 PM

*Option #3 - Allows the user to specify what specific information the user intends to search. (Note - This is not an extensive search engine and can handle basic searches)*

**Chat History**

[21:39:38] 🤖 Patent Analysis Assistant

Please select an option:

1. 📊 Analyze existing patent
2. 💡 Analyze new invention
3. 🔎 Search for similar patents

Enter 1, 2, or 3:

[21:39:41] 3

[21:39:41] 🔎 What technology or topic would you like to search for?  
(Examples: machine learning, IoT, blockchain, etc.)

[21:39:59] Pattern Recognition

*Searches the RAG DB to identify any similar patents that have “Pattern Recognition” as the key. If the program is able to find a minimum of 8 patents in the RAG DB, it will publish those results. If the number of similar patents in RAG is less than 8 then for the delta patents it will conduct a patent search on Google Patent Dataset.*

**Chat History**

[21:39:59] 🔎 PATENT SEARCH RESULTS (RAG Database)

📚 COMPREHENSIVE PATENT SUMMARY:

**Summary of Pattern Recognition Patents**

1. **Patent Number:** US9471759B2  
**Inventors:** Alexander Kott et al.  
**Short Description:** This patent describes a system and method for identifying patterns in time-series data to detect anomalies or unusual events within datasets applicable across finance, healthcare, and cybersecurity.  
**Key Innovation/Technology:** Machine learning techniques for anomaly detection.

2. **Patent Number:** US9189576B2  
**Inventors:** John Canny et al.  
**Short Description:** This patent outlines a method using pattern recognition to improve network security by identifying normal behavior patterns and detecting deviations indicating malicious activities.  
**Key Innovation/Technology:** Network security enhancement through behavioral analysis.

3. **Patent Number:** US8543602B1

 Chat History ↗ prices or other financial indicators using advanced statistical models and machine learning techniques.



**Key Innovation/Technology:** Financial prediction through pattern recognition.

8. **Patent Number:** US10354279B2

**Inventors:** Christopher W. Jones et al.

**Short Description:** This patent describes a system recognizing patterns in medical imaging data to assist radiologists in diagnosing diseases more accurately and efficiently using deep learning algorithms.

**Key Innovation/Technology:** Deep learning for medical image analysis.

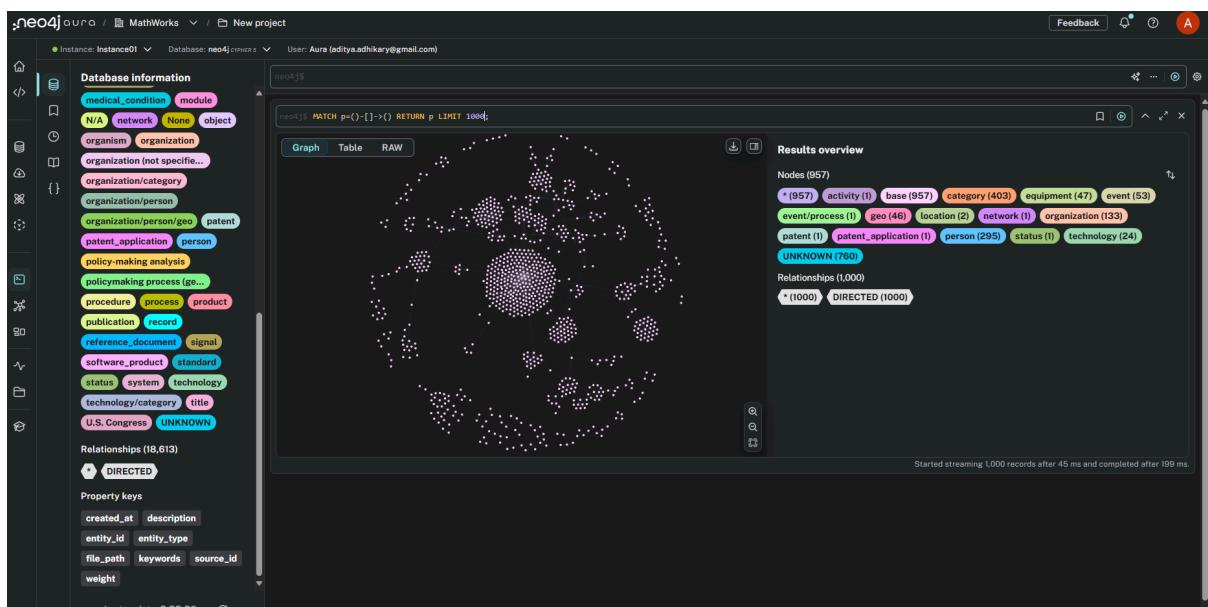
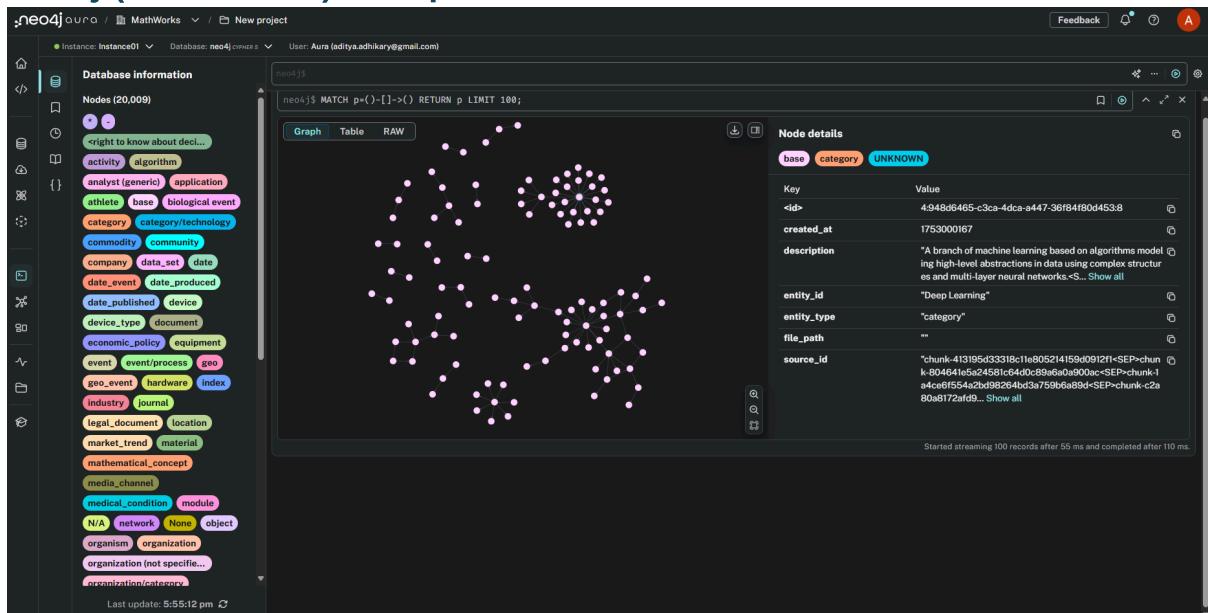
This list comprises eight patents that closely relate to pattern recognition across different technological domains including cybersecurity, distributed computing, Internet of Things (IoT), social networks, user interface personalization, financial markets, and medical imaging diagnostics.

 SEARCH SUMMARY:

- Query: "Pattern Recognition"
- RAG Database: Found 8 relevant patents
- Source: LightRAG patent database



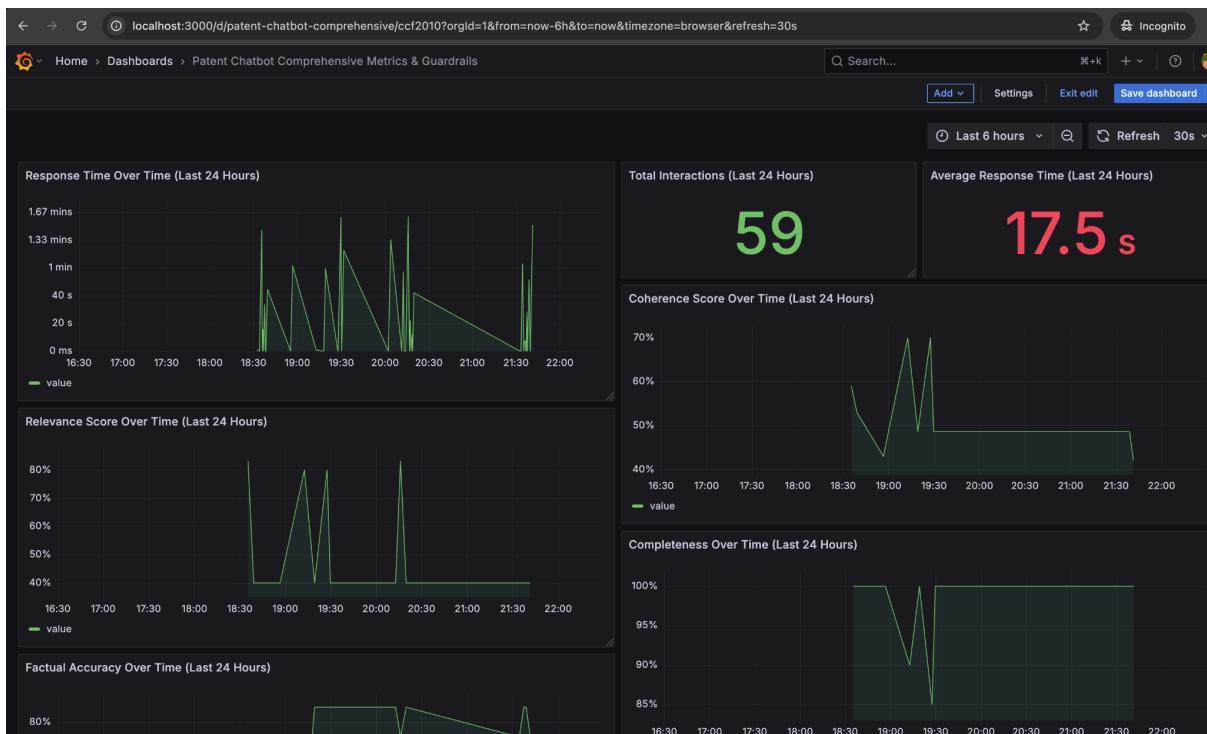
## Neo4j (Neo4J Aura) - Graph Visualizations

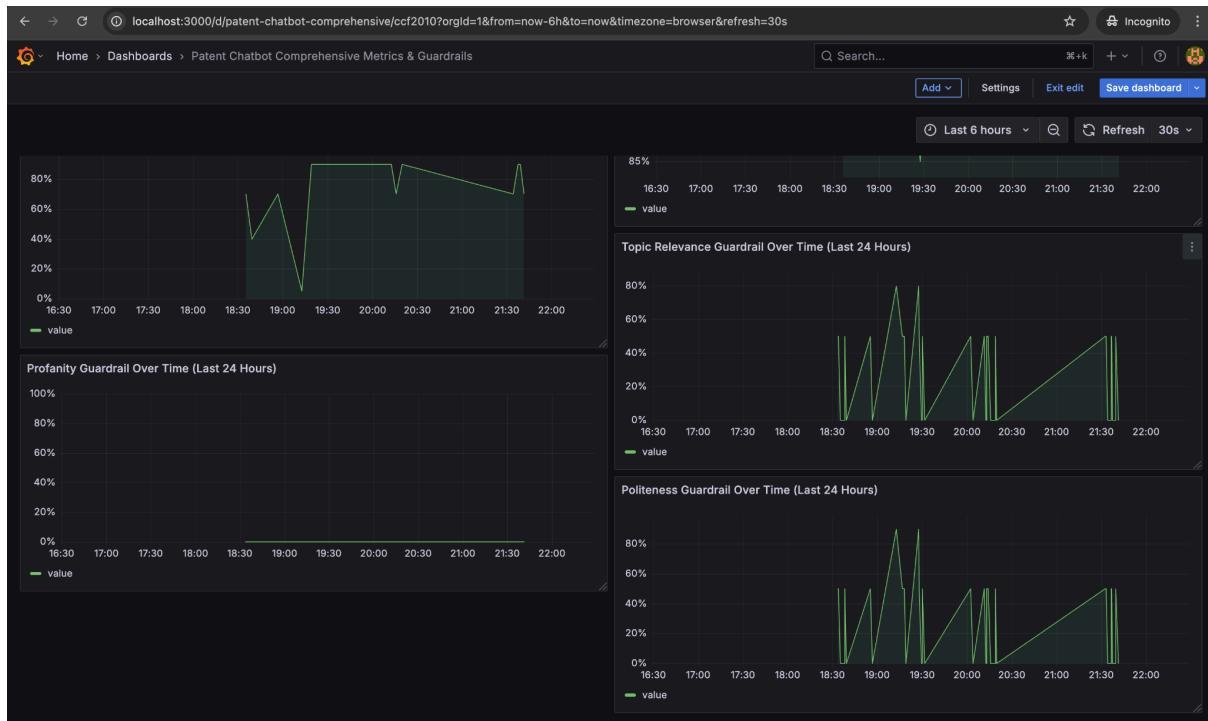


## VectorDB (Postgres Serverless - Neon)

Screenshot of a PostgreSQL database interface showing the `test-pgvector` schema. The `Tables` section lists various tables including `neondb`, `public`, `lightrag_doc_chunks`, `lightrag_doc_full`, and `lightrag_doc_status`. The main view displays a table with columns: `workspace`, `id`, `content_text`, `content_summary`, `content...`, `chunks...`, `status`, `file_pat...`, and `chunks_list`. The table contains approximately 50 rows of document data with their respective IDs and processing status.

## Monitoring System





## Appendix

**Source Code:** <https://github.com/aniketrastrasti1987/capstone>

## References

1. United States Patent Dataset (HUPD)
2. Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, Chao Huang - [LightRAG: Simple and Fast Retrieval-Augmented Generation](#)
3. Lewis, P., Oguz, B., Rinott, R., Schick, T., Dwivedi-Yu, J., Khashabi, D., & Riedel, S. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. arXiv preprint arXiv:2005.11401.
4. Neo4j: <https://neo4j.com>
5. HuggingFace Transformers:  
<https://huggingface.co/docs/transformers/index>
6. Ollama Project: <https://ollama.com>
7. GPT-4 by OpenAI: <https://openai.com/gpt-4>
8. BGE-M3 Embeddings: <https://ollama.com/library/bge-m3>

9. Qwen 2.5 14B- Instruct:

<https://ollama.com/library/qwen2.5:14b-instruct>

10. Cohere Reranker 3.5: <https://docs.cohere.com/docs/rerank-overview>

11. LightRAG Code: <https://github.com/HKUDS/LightRAG>