Agentic AI + LLM powered document information retrieval system

Abstract

Regular, general-purpose LLM models such as GPT use it’s training data, which is often scraped from the internet, to answer queries from the user. And as we could imagine, an answer that is correct at one point in time may not be correct at another. This is where these general-purpose LLM models fail. They fail to provide answers, advice or make decisions based on real-time data.

Hence, when a problem such as creating a system that uses LLMs to process natural language queries and retrieve relevant information from large, unstructured documents such as policy documents, contracts and emails presents itself, a few issues that are inherent to LLMs comes to mind – such as LLM hallucinations, missing key information (even in systems with semantic search, since it’s possible that some information is semantically far apart but also is crucial information to answer the user’s query) and providing outdated advice based on the LLM system’s training data.

This document explores a possible solution that overcomes the aforementioned issues to a high degree, making use of recent technological advancements such as agentic AI and RAG, along with some classic, well-known LLM techniques such as end-to-end evaluation, relevance assessment, knowledge graphs and more. This solution makes sure that the system “understands” the document at a human-level to the highest degree possible.

Introduction

Analysing unstructured data has been a persistent challenge in data processing. Researchers have been trying to solve the problem of giving machines the ability to perceive and understand the world like humans do for as early as 1959, which is when David Hubel and Torsten Wiesel released their work “Receptive fields of single neurons in the cat’s striate cortex” which showed the existence of simple and complex neurons. This was one of the pivotal moments that contributed to the development of machine learning concepts such as CNN and RNN. When it comes to LLMs, it started in the early 1990s when IBM’s statistical models pioneered word alignment techniques for machine translation, which now is the simplest form of a language model.

A Large Language Model (LLM) is a language model trained with self-supervised machine learning on a vast amount of text and is designed for Natural Language Processing (NLP) tasks, especially language generation, as seen with generative AI in recent years. These models, when trained with sufficient data, acquire the ability to predict and generate syntax, semantics and ontologies in our natural language. However, these systems lack logic, and biases present in the training data are introduced, especially when the LLM works on a simple concept such as the n-gram model. This happens since when large amounts of data is required, it’s hard to make sure that the data is fairly divided between all topics and subtopics. This issue is also worsened when LLMs such as GPT acquire most of their training data from the internet.

This alone introduces two issues in LLM systems:

1) The inherent biases introduced due to the LLM’s training data and
2) All information provided are based on the LLM’s training data, which is not time-aware – hence the information provided by an LLM may be incorrect for a given time since the data its using to answer may be out-of-date, without the LLM knowing.

These two shortcomings become detrimental when an LLM system needs to perform real-world tasks and make time and information specific decisions, such as answering queries related to insurance, contracts, etc.

The main goal for this system is to be able to parse natural language, vague user queries, then search and retrieve relevant clauses or rules from the provided documents using semantic understanding rather than simple keyword matching. This information must then be evaluated to arrive at the correct decision based on the user’s query, and this answer must be returned as a structured JSON response that can be used by downstream systems. LLMs alone won’t be sufficient to arrive to this goal. So, in addition to LLMs, this solution will integrate the use of agentic AI and other complementary LLM techniques for each task.

Agentic AI is an autonomous, goal-oriented AI system that overcomes the shortcomings of general-purpose LLM models by being able to make autonomous decisions based on real-time data. It can work with multiple agents with the help of AI orchestration. Agentic AI builds on generative AI techniques by using large language models to function in dynamic environments. Agents can specialize in a specific task. This is one of the advantages of agentic AI over general-purpose AI such as GPT, and developers can use this advantage in various levels of development of a system. For instance, in an LLM-powered intelligent query retrieval system, dividing the main tasks of parsing, reading, understanding the input document and answering the user’s query can all be divided into smaller, specialized tasks that can be handled by various agents that work together. Also, other systems/downstream agents can also easily work with this agentic system as a whole, making this LLM-powered intelligent query retrieval system a pluggable component to other domain specific systems.

Problem Statement

The system needs to use Large Language Models (LLMs) to process natural language queries and retrieve relevant information from large unstructured documents such as policy documents, contracts and emails, and provide a time-sensitive and information-specific decision or advice for the user based on their query.

The following are the objectives of this system, assuming an input query such as:

“30-year-old male, knee surgery, 3-month-old insurance policy.”

Then the system must:

- Parse and structure the query to identify key details such as age, procedure, location, and policy duration.

- Search and retrieve relevant clauses or rules from the provided documents using semantic understanding rather than simple keyword matching.

- Evaluate the retrieved information to determine the correct decision, such as approval status or payout amount, based on the logic defined in the clauses.

- Return a structured JSON response containing: Decision (e.g., approved or rejected), Amount (if applicable), and Justification, including mapping of each decision to the specific clause(s) it was based on.

- Be capable for application in domains such as insurance, legal compliance, human resources and contract management.

A sample response may be:

"Yes, knee surgery is covered under the policy."

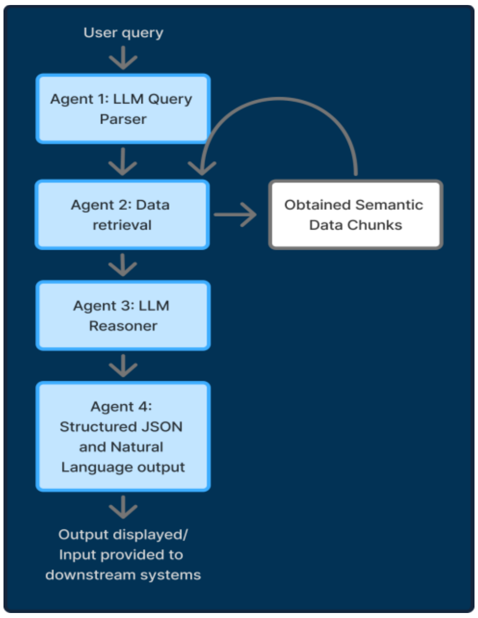
User query
Agent 1: LLM Query Parser
Agent 2: Data retrieval
Obtained Semantic Data Chunks
Agent 3: LLM Reasoner
Agent 4: Structured JSON and Natural Language output
Output displayed/ Input provided to downstream systems

Fig 1: Process flow diagram of the Agentic AI + LLM powered information retrieval system.

Related Work

This section contains a literature survey conducted on journal and conference articles related to the problem statement written in English and published after 2021. This is an explorative literature survey that was conducted to learn more about the problem statement, technologies involved and existing solutions to similar problems, and hence helped arrive at an actionable solution to the problem at hand.

Article 1: DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. (2025) Published in arxiv.

Authors: Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, Eugene Wu.

Objective: This article presents DocETL, a system that optimizes complex document processing pipelines, while accounting for LLM shortcomings. The LLM shortcomings identified in this article:

- LLMs trade accuracy for reduced cost when executing user specific operations by expressing tasks as a single-step map operation.
- Outputs are, as a result, inaccurate even with optimized prompts, since these systems assume user-defined operations will yield sufficiently accurate results when executed by the LLM, and if the document in question exceeds the LLM's context limit, results provided will be inaccurate due to LLM hallucinations.
- Even if the document fits this limit, the LLM may miss key information or include spurious details.

Methodology: The DocETL application introduces:

- Logical rewriting of pipelines tailored for LLM-based tasks
- Agent-guided plan evaluation mechanism that synthesizes and orchestrates task specific validation prompts
- Optimization algorithm that efficiently finds promising plans, considering the latencies of agent-based plan generation and evaluation.

These are done through a YAML-based interface for users to create and author their pipelines with LLM-specific operators, including "resolve" for entity resolution, and "gather" to maintain context when processing document chunks. Users can specify their pipelines through this application while the application decomposes, rewrites and optimizes the pipeline.

Limitations: While DocETL is said to demonstrate accuracy, thereby showing improvements in real-world applications in comparison to traditional LLM systems, it is acknowledged that the system still consists of biases, even though the are reduced significantly using the perspectives of different agents and using their optimizer to further optimize the model.

This system also consists a YAML-based user interface, that may not be usable by most casual users with a non-technical background. This system provides a novel approach for rewriting the single-step LLM pipeline to match the user's requirements. However, this system would not work if the user expects actionable advice and decisions by providing a query in natural language that may or may not be vague, and one or many unstructured documents.

Article 2: Revisiting Uncertainty-based Query Strategies for Active Learning with Transformers.(2022) Published in arxiv.

Authors: Christopher Schröder, Andreas Niekler, Martin Potthast.

Objectives: This paper systematically explores various uncertainty-based query strategies as a computationally inexpensive alternative, despite their disadvantages compared to other recent active learning techniques. It accesses the performance of various combinations of uncertainty-based query strategies along with transformer models such as BERT and DistilRoBERTa.

Observations: Five uncertainty-based query strategies in combination with the aforementioned transformer models, and the investigation showed that these strategies still perform well. These strategies, being computationally inexpensive, can be a great addition to the techniques that this solution will use to parse and rewrite user queries so that the LLM model understands the user query, and hence be able to look for the correct information from the document.

Article 3: Context Aware Query Rewriting for Text Rankers using LLM.(2023) Published in arxiv.

Authors: Abhijit Anand, Venktesh V, Vinay Setty, Avishek Anand.

Objectives: This article explores the limitations of LLMs when it comes to query rewrites and how to utilize LLMs to improve query rewriting for text ranking tasks. They use LLM-based query rewriting only during the training phase, and use context-aware query rewriting to leverage the benefits of LLMs for query understanding.

This is different from traditional LLM systems since they use LLM-based query rewriting for the main systems as well and do not use context-aware techniques.

Methodology: The Context Aware Rewriter (CAR) framework reformulates ambiguous and vague user queries by employing a context-aware prompting of an LLM. A ranker model fine-tuned on disambiguated queries is directly employed to rank new queries without rewriting them. The query rewrites are generated by conditioning the LLM through few-shot prompting on the relevant document for the query to avoid drift.

“This context-aware prompting of LLMs results in better rewrites without topic drifts, as the LLM is conditioned on intents conveyed in the document. We also introduce a constraint on the maximum length of the output sequence generated.”

Limitations: This method identifies and chooses ambiguous queries based on heuristics like query length and specific types of queries with multiple meanings based on context from the document. This method can be improved further by improving the heuristic considered.

Takeaways: The CAR framework provides a specific and well thought out framework that can be used to reformulate vague queries without drifting away from what the user probably meant, using context from the document provided by the user. This framework can be used along with the uncertainty-based query strategies to overcome the limitations that are introduced by heuristics.

Article 4: Query Understanding in LLM-based Conversational Information Seeking.

Authors: Yifei Yuan, Zahra Abbasiantaeb, Mohammad Aliannejadi, Yang Deng.

Objectives: This article explores LLM-driven methods for developing evaluation metrics to assess query understanding quality in multiturn interactions, strategies for building more interactive systems and more. Out of these objectives, the most relevant objective is the query understanding evaluation. The system interprets the intent and context of a user’s query to deliver more accurate and relevant search results.

Methodology: The two main evaluation techniques used are end-to-end evaluation and LLM-based relevance assessment. End-to-end evaluation uses human-judged benchmarks to assess the relevance of query-answer pairs. These benchmarks include QReCC and TopioCQA. Relevance assessment leverages LLMs to evaluate the relevance of the retrieved information to a user’s query.

Limitations: One of the persisting challenges highlighted in this article is with multilingual and cross-cultural query understanding, since these methods are specific to the English language. To overcome this challenge, techniques that cater to multilingual queries need to be implemented in the future.

Article 5: Large Language Model Powered Agents for Information Retrieval.

Authors: An Zhang, Yang Deng, Yankai Lin, Xu Chen, Ji-Rong Wen, Tat-Seng Chua.

Objectives: This article presents a comprehensive exploration of how LLM-powered agents can enhance various information retrieval (IR) applications, including search engines, recommender systems, social networks, and conversational assistants. The primary objective is to provide an in-depth overview of cutting-edge designs for such agents, discuss the open challenges they face, and identify potential research directions that could revolutionize IR. The tutorial also seeks to foster discussions that extend beyond IR, touching upon areas such as human-computer interaction and computational social science.

Methodology: The paper synthesizes recent research and real-world deployments, distilling common architectural patterns for LLM-powered agents into a unified framework comprising profiling, memory, planning, and action modules. It examines how these agents are applied in different domains—such as simulating social networks, solving domain-specific problems, improving recommender systems through personalized interaction, and enabling conversational agents capable of user simulation, proactive dialogues, and multi-turn instruction following. The analysis highlights integration strategies and use cases, illustrating how LLMs' human-like reasoning and adaptability can address traditional IR system limitations.

Limitations: Some of the acknowledged limitations are from the current state of LLM-powered agents. The system risks hallucinations, inconsistencies and ethical issues that is caused due to the training data. In our solution, this problem can be mitigated by creating document models that cover the various semantic and logical links between information present in documents, and dividing high computation tasks between multiple agents to avoid hallucinations.

Proposed Solution

This suggested solution is an **LLM-powered agentic system** that understands vague natural language queries, reasons over unstructured documents and returns traceable, structured decisions with clear justifications.

This system works on an **agent-based approach**, rather than using general-purpose LLM models such as GPT since they provide answers based on its training data, and we do not want advice/decisions from the past, we want **decisions that consider real-time information** – and an agentic AI system powered by LLM orchestration is exactly what's needed for that. This system will contain multiple agents to:

- Parse and rewrite vague, natural language queries and extract structured information from the query with the help of an agent powered with uncertainty-based query strategies, end-to-end evaluation and relevance assessment.
- Perform semantic search with the help of various advanced NLU techniques. To avoid missing relevant pieces of information, the agent retrieves relevant chunks **iteratively**, ensuring that the system acquires a better level of understanding of the document.
- The retrieved data is now used to arrive at a decision. Agentic AI help with the reasoning process using RAG systems and multiagent reasoning.
- Final output is structured as a JSON for downstream systems and a natural language output for the user.

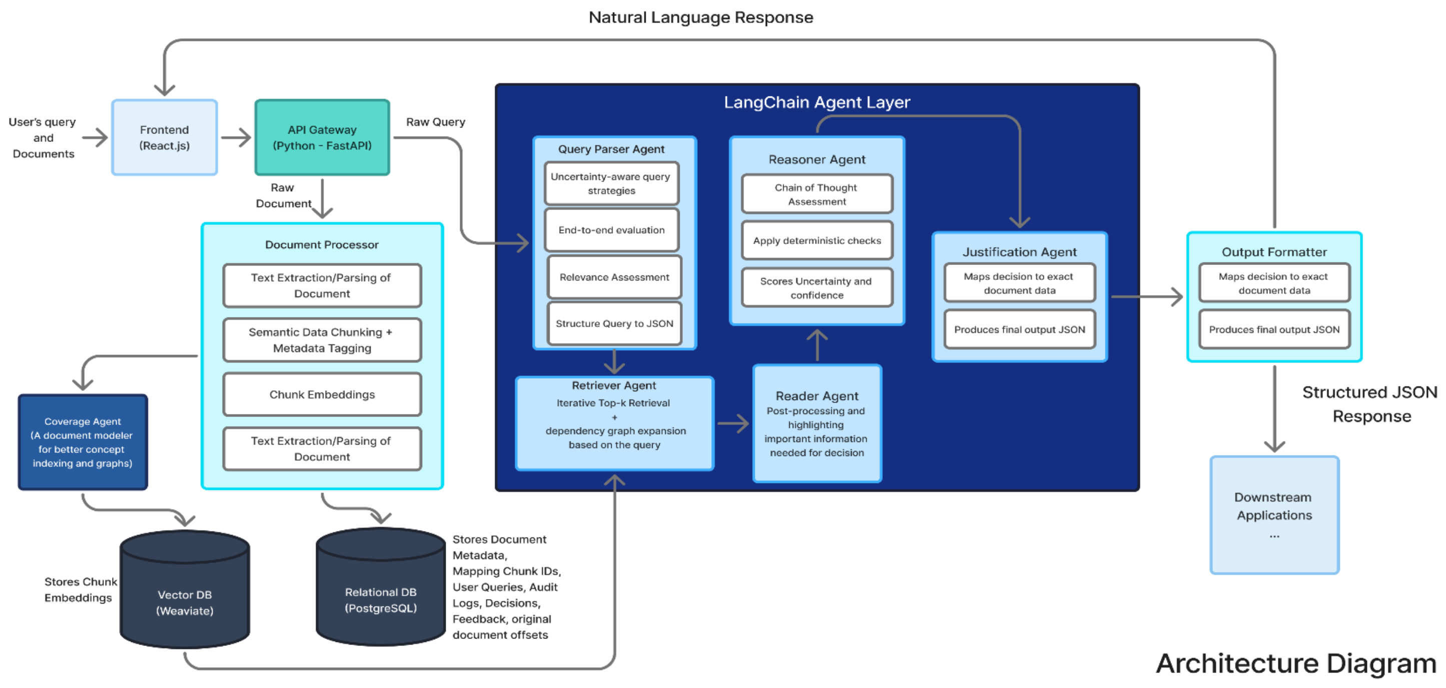
Natural Language Response
User's query and Documents
Frontend (React.js)
API Gateway (Python - FastAPI)
Raw Query
Raw Document
Document Processor
Text Extraction/Parsing of Document
Semantic Data Chunking + Metadata Tagging
Chunk Embeddings
Text Extraction/Parsing of Document
Coverage Agent (A document modeler for better concept indexing and graphs)
Stores Chunk Embeddings
Vector DB (Weaviate)
Relational DB (PostgreSQL)
Stores Document Metadata, Mapping Chunk IDs, User Queries, Audit Logs, Decisions, Feedback, original document offsets
LangChain Agent Layer
Query Parser Agent
Uncertainty-aware query strategies
End-to-end evaluation
Relevance Assessment
Structure Query to JSON
Reasoner Agent
Chain of Thought Assessment
Apply deterministic checks
Scores Uncertainty and confidence
Retriever Agent
Iterative Top-k Retrieval + dependency graph expansion based on the query
Reader Agent
Post-processing and highlighting important information needed for decision
Justification Agent
Maps decision to exact document data
Produces final output JSON
Output Formatter
Maps decision to exact document data
Produces final output JSON
Structured JSON Response
Downstream Applications ...
Architecture Diagram

Fig 2: Architecture diagram of the Agentic AI + LLM powered information retrieval system.

The following table briefly explains each high-level component used in this solution:

Table 1: Description of high-level components used in the Agentic AI + LLM powered information retrieval system.

Component	Explanation
Frontend and API Gateway	React.js frontend for entering query, uploading documents and displaying output. FastAPI backend (Python) contains endpoints such as /query, /upload. It also orchestrates LangChain tool calls.
Document Processing	The uploaded document is parsed using PyMuPDF/docx2txt. This parsed data undergoes semantic chunking and metadata tagging. Embedding is produced for each of these chunks.
Coverage Agent	This agent is invoked at the document processing time to create a multi-layered document model such as a clause graph, concept index, entailment/contradiction matrix, hierarchical embeddings and summaries and more. The goal of this agent is to model the document so that semantic retrieval rarely misses important, logical context.
Vector Database	The enhanced embeddings from the coverage agent is stored in the vector database.
Relational Database	A Postgres database stores metadata, mapping chunk IDs, user queries, audit logs, decisions and feedback.
LangChain Agent Layer	This layer contains multiple agents that are responsible for various tasks. • Query Parser Agent: Parses and rewrites vague natural language queries into structured JSON with entities that scored the highest in the relevance assessment. • Retriever Agent: Retrieves data from the vector database using various techniques such as an iterative top-k retrieval and dependency graph expansion. • Reader Agent: Reads retrieved chunks of data and highlights important information. • Justifier Agent: This agentic AI agent makes autonomous decisions based on the data read by the reader agent and maps

	it with that data, and stores the mapping chunk IDs and user query to the relational database. • Controller Agent: Orchestrates the sequence of the working of agents, handles uncertainty and makes decisions on if clarifying questions needs to be asked to the user

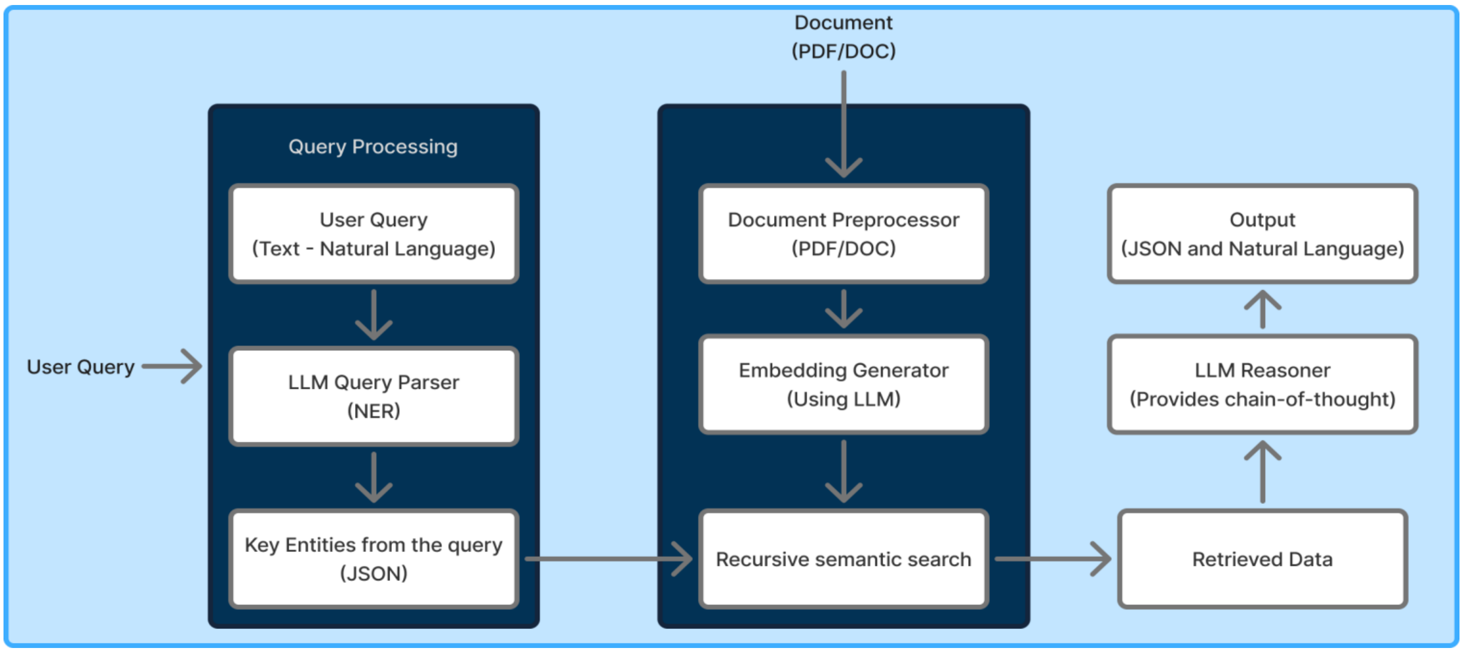

Fig 3: Data flow diagram of the Agentic AI + LLM powered information retrieval system.

Technological Specifications

Hardware Specifications

- Intel Core i5 2nd generation is a minimum requirement for this system. A better processer can be used as well.
- 4 to 8 GB RAM is ideal.
- 40MB of free hard disk drive space for the smooth running of the hardware system and this application.
- 1024x768 monitor resolution.

Software Specifications

- Windows 10 or higher for an operating system.
- Visual Studio Code as the code editor.
- Python version 3.10 or higher
- Cloud Service Providers : Google Cloud Platform, Vercel and Render for deployment of the application.
- Database : Pinecone for a vector database and PostgreSQL for a relational database.
- Backend : Python – FastAPI, LangChain to create an Agentic AI agents layer, Oauth for authentication, FAISS for semantic search, GPT-4 for the LLM.
- Frontend: React.js for quick prototyping and to provide a simple and effective frontend.
- Others: PyMuPDF, Apache Tika for document pre-processing, GitHub for version control.

Conclusion

The proposed LLM-powered intelligent query retrieval system addresses the critical shortcomings of conventional general-purpose language models by integrating agentic AI, Retrieval-Augmented Generation (RAG), and specialized query processing techniques. Unlike traditional LLMs, which rely solely on static training data, this system enables time-aware, document-specific reasoning over large, unstructured datasets such as contracts, policy documents, and compliance records.

By employing multiple specialized agents—each tasked with parsing, semantic retrieval, reasoning, and justification—the architecture ensures that vague, natural language queries are transformed into precise, traceable decisions supported by relevant document clauses. The use of uncertainty-based query strategies, multi-layered document modelling, and iterative retrieval further minimizes the risks of missing semantically distant but crucial information.

This multi-agent, orchestration-driven approach not only enhances retrieval accuracy and interpretability but also outputs decisions in both structured JSON for downstream integration and natural language for user readability. The result is a flexible, pluggable component capable of seamless adoption across domains such as insurance, legal compliance, human resources, and contract management.

In essence, this system bridges the gap between human-level comprehension and machine processing, offering a scalable, transparent, and domain-agnostic solution to complex, time-sensitive decision-making over unstructured documents.

References

1) Shankar, S., Chambers, T., Shah, T., Parameswaran, A. G., & Wu, E. (2025, April 1). *DocETL: Agentic query rewriting and evaluation for complex document processing* (arXiv:2410.12189) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2410.12189
2) Schröder, C., Niekler, A., & Potthast, M. (2022, March 20). *Revisiting uncertainty-based query strategies for active learning with transformers* (arXiv:2107.05687) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2107.05687
3) Anand, A., V., V., Setty, V., & Anand, A. (2023, August 31). *Context aware query rewriting for text rankers using LLM* (arXiv:2308.16753) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2308.16753
4) IBM. (2025). *The 2025 guide to AI agents*. IBM. https://www.ibm.com/think/ai-agents#605511093
5) Caballar, R. D., & Stryker, C. (2025). *AI agent frameworks: Choosing the right foundation for your business*. IBM. https://www.ibm.com/think/insights/top-ai-agent-frameworks
6) Yuan, Y., Abbasiantaeb, Z., Aliannejadi, M., & Deng, Y. (2025, July 13). *Query understanding in LLM-based conversational information seeking*. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 4098–4101). ACM. https://doi.org/10.1145/3726302.3731687
7) Zhang, A., Deng, Y., Lin, Y., Chen, X., Wen, J., & Chua, T.-S. (2024). Large language model powered agents for information retrieval. *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*, 2989–2992. https://doi.org/10.1145/3626772.3661375
8) Hariharan, M. (2025, April). *Semantic Mastery: Enhancing LLMs with Advanced Natural Language Understanding* (arXiv:2504.00409). arXiv. https://arxiv.org/abs/2504.00409
9) Lyre, H. (2024, February). *"Understanding AI": Semantic grounding in large language models* (arXiv:2402.10992). arXiv. https://arxiv.org/abs/2402.10992
10) Shruthi, K., Tulasi, E., & Anjali, R. (n.d.). *AI powered document processing system using LangChain & semantic search*. Department of Information Technology, Anurag University, Hyderabad, Telangana.
11) Pugliese, R., Kourousias, G., Venier, F., & Garlatti Costa, G. (2025, May). *Agentic Publications: An LLM-driven framework for interactive scientific publishing, supplementing traditional papers with AI-powered knowledge systems* (arXiv:2505.13246). arXiv. https://doi.org/10.48550/arXiv.2505.13246

12) Raza, S., Sapkota, R., Karkee, M., & Emmanouilidis, C. (2025, June). *TRiSM for Agentic AI: A review of trust, risk, and security management in LLM-based agentic multi-agent systems* (arXiv:2506.04133; Version 3, submitted July 9, 2025). arXiv. https://doi.org/10.48550/arXiv.2506.04133

13) Khalil, R. A., Ahmad, K., & Ali, H. (2025, July). *Redefining elderly care with Agentic AI: Challenges and opportunities* (arXiv:2507.14912). arXiv. https://doi.org/10.48550/arXiv.2507.14912

14) Hoseini, S., Burgdorf, A., Paulus, A., Meisen, T., Quix, C., & Pomp, A. (2024). *Towards LLM-augmented creation of semantic models for dataspaces.* In *Proceedings of the Second International Workshop on Semantics in Dataspaces* (Vol. 3705, Paper 03). CEUR-WS.