Project 2: InsightHub - A Customizable RAG System

Nate Stott

December 9, 2025

Contents

1 Introduction

Welcome to InsightHub, a configurable and flexible system that allows users to create any RAG System they would like. Every step of the RAG process can be picked and changed to any preference. Can't find an option you would like? The application has been made so it's easy to extend. Simply go to the step folder you would like to add an addition option for, and implement what you would like from the steps interface. Currently, InsightHub only offers a CLI (this is a MVP product currently) but in the future a scalable website will be featured. All the core features talked about in the proposal report are implemented and well tested.

The rest of the report will detail what the application can do. See the main README for instructions about how to get setup and going if you want to use the application.

Also keep in mind there are some good docs in the docs folder that go into deeper detail from a users and a developers prospective. There are helpful tutorial docs, start at docs/users/tutorials/cli/README.md to complete the tutorals if you want a play-by-play example.

You can use the `--help` flag anywhere in the command structure for further details.

```
$ task cli -- --help
usage: insighthub [-h] {workspace,document,chat,default-rag-config,state,rag-options} ..

InsightHub CLI - Chat-centric RAG interface

positional arguments:
  {workspace,document,chat,default-rag-config,state,rag-options}
                        Resource to manage
    workspace           Workspace operations
    document            Document operations
    chat                Chat operations
    default-rag-config  Default RAG configuration
    state               Application state
    rag-options         RAG options

options:
  -h, --help            show this help message and exit

Examples:
  # Workspace management
  python -m src.cli workspace list              List all workspace
  python -m src.cli workspace create            Create a new workspace (interactive)
  python -m src.cli workspace select 1          Select workspace with ID 1

  # Document management
  python -m src.cli document list               List document in current workspace
  python -m src.cli document show 1             Show detailed information about document
  python -m src.cli document add file.pdf       Add a document to current workspace
  python -m src.cli document remove file.pdf    Remove a document from current workspace

  # Chat operations
  python -m src.cli chat list                   List chat sessions in current workspace
  python -m src.cli chat create                 Create a new chat session in current wor
  python -m src.cli chat select 1               Select chat session with ID 1
  python -m src.cli chat send "Hello"           Send a message to current session
  python -m src.cli chat history                Show message history for current session
  python -m src.cli chat delete 1               Delete a chat session

  # Configuration
```

```
  python -m src.cli default-rag-config show     Show default RAG configuration
  python -m src.cli default-rag-config create   Create/update default RAG config (intera

  # State
  python -m src.cli state show                  Show current state (selected workspace/s

  # RAG Options
  python -m src.cli rag-options list            List all available RAG options

For help on a specific resource, use:
  python -m src.cli <resource> --help
$
```

Keep in mind `task cli --` is just an alias for `poetry run python -m src.cli` (see Taskfile.yml) and `$` denotes the terminal is ready to except user input.

2 Core Features

At the core of the application is the notion of a workspace. A workspace has a rag config, documents, and chat sessions. The rag config for a workspace is immutable once the workspace is created, but you can add, remove and edit documents and chat sessions. If you want a different rag config, no worries just make a new workspace.

```
$ task cli -- workspace list
No workspace found
$ task cli -- workspace create
Workspace name: BOSCH BMV080
Description (optional):

Available RAG types:
  - vector: Traditional vector similarity search with embeddings
  - graph: Knowledge graph-based retrieval with entity relationships

RAG type [vector]:

Available chunking algorithms:
  - sentence: Split by sentence boundaries
  - character: Fixed character size chunks
  - semantic: Semantically coherent chunks
  - token: Split by token boundaries
  - markdown: Split by Markdown sections
  - html: Split by HTML elements
  - code: Split by code structures

Available embedding algorithms:
```

```
  - nomic-embed-text: Nomic AI embedding model (274M params)
  - all-MiniLM-L6-v2: Sentence-BERT embedding model
  - mxbai-embed-large: Large multilingual embedding model

Available reranking algorithms:
  - none: Return results as-is from vector search
  - cross-encoder: Rerank using cross-encoder model for better relevance
  - bm25: Rerank using BM25 algorithm for keyword-based relevance
  - rrf: Combine multiple rankings using RRF algorithm

Chunking algorithm [sentence]:
Chunk size [1000]:
Chunk overlap [200]:
Embedding algorithm [nomic-embed-text]:
Top K [5]:
Rerank algorithm [none]:
Created workspace [1] BOSCH BMV080
$ task cli -- workspace list
[1] BOSCH BMV080
$ task cli -- workspace select 1
Selected [1] BOSCH BMV080
$ task cli -- workspace list
[1] BOSCH BMV080 (SELECTED)
$
```

Once your workspace is ready, you can upload documents which depending on your rag config could take some time to process. Once your document or documents are done processing, you can open a chat session and ask questions about the documents. The LLMs responses will be automatically infused with the proper context to properly answer your questions.

```
$ task cli -- chat list
No chat sessions found
$ task cli -- chat create
Session title (optional): no context
Created chat session [1] no context
$ task cli -- chat list
[1] no context
$ task cli -- chat select 1
Selected [1] no context
$ task cli -- chat list
[1] no context (SELECTED)
$ task cli -- chat send "What is the maximum power consumption of the BMV080 in continuc

You: What is the maximum power consumption of the BMV080 in continuous measurement mode?

Assistant: I can't provide information on the maximum power consumption of a specific de
```

```
$ task cli -- chat create
Session title (optional): with context
Created chat session [2] with context
$ task cli -- chat list
[2] with context
[1] no context (SELECTED)
$ task cli -- chat select 2
Selected [2] with context
$ task cli -- document add /home/nate/Downloads/bmv080-ds.pdf
Adding bmv080-ds.pdf...
Added [1] bmv080-ds.pdf
$ task cli -- document list
[1] bmv080-ds.pdf
$ task cli -- chat send "What is the maximum power consumption of the BMV080 in continuc

You: What is the maximum power consumption of the BMV080 in continuous measurement mode?

Assistant: The maximum power consumption of the BMV080 in continuous measurement mode is
$
```

3 Other Features

There are some additional minor quality of life features. The default rag config will be the starting place when creating a new workspace.

```
$ task cli -- default-rag-config show
RAG Type: graph
Entity Extraction: spacy
Relationship Extraction: dependency-parsing
Clustering Algorithm: leiden
$ task cli -- default-rag-config create

Available RAG types:
  - vector: Traditional vector similarity search with embeddings
  - graph: Knowledge graph-based retrieval with entity relationships

RAG type [vector]: graph

Available entity extraction algorithms:
  - spacy: Fast and accurate named entity recognition using spaCy's transformer models.
  - llm: Flexible entity extraction using large language models with structured JSON out

Available relationship extraction algorithms:
  - dependency-parsing: Extract relationships using spaCy's dependency parser to identif
  - llm: Flexible relationship extraction using large language models with structured JS
```

```
Available clustering algorithms:
  - leiden: Advanced community detection using the Leiden algorithm. Improved quality ov
  - louvain: Fast community detection using the Louvain algorithm. Good for large graphs

Entity extraction algorithm [spacy]:
Relationship extraction algorithm [dependency-parsing]: llm
Clustering algorithm [leiden]: louvain
Default RAG config saved
$ task cli -- default-rag-config show
RAG Type: graph
Entity Extraction: spacy
Relationship Extraction: llm
Clustering Algorithm: louvain
$ task cli -- default-rag-config create

Available RAG types:
  - vector: Traditional vector similarity search with embeddings
  - graph: Knowledge graph-based retrieval with entity relationships

RAG type [vector]:

Available chunking algorithms:
  - sentence: Split by sentence boundaries
  - character: Fixed character size chunks
  - semantic: Semantically coherent chunks
  - token: Split by token boundaries
  - markdown: Split by Markdown sections
  - html: Split by HTML elements
  - code: Split by code structures

Available embedding algorithms:
  - nomic-embed-text: Nomic AI embedding model (274M params)
  - all-MiniLM-L6-v2: Sentence-BERT embedding model
  - mxbai-embed-large: Large multilingual embedding model

Available reranking algorithms:
  - none: Return results as-is from vector search
  - cross-encoder: Rerank using cross-encoder model for better relevance
  - bm25: Rerank using BM25 algorithm for keyword-based relevance
  - rrf: Combine multiple rankings using RRF algorithm

Chunking algorithm [sentence]: semantic
Chunk size [1000]:
Chunk overlap [200]:
```

```
Embedding algorithm [nomic-embed-text]:
Top K [5]:
Rerank algorithm [none]: bm25
Default RAG config saved
$ task cli -- default-rag-config show
RAG Type: vector
Chunking Algorithm: semantic
Chunk Size: 512
Chunk Overlap: 100
Embedding Algorithm: nomic-embed-text
Top K: 3
Rerank Algorithm: bm25
$
```

You can also check out what rag options are available with descriptions of each step.

```
$ task cli -- rag-options list
Available RAG Options:

RAG Types:
  - vector: Traditional vector similarity search with embeddings
  - graph: Knowledge graph-based retrieval with entity relationships

Chunking Algorithms:
  - sentence: Split by sentence boundaries
  - character: Fixed character size chunks
  - semantic: Semantically coherent chunks
  - token: Split by token boundaries
  - markdown: Split by Markdown sections
  - html: Split by HTML elements
  - code: Split by code structures

Embedding Algorithms:
  - nomic-embed-text: Nomic AI embedding model (274M params)
  - all-MiniLM-L6-v2: Sentence-BERT embedding model
  - mxbai-embed-large: Large multilingual embedding model

Rerank Algorithms:
  - none: Return results as-is from vector search
  - cross-encoder: Rerank using cross-encoder model for better relevance
  - bm25: Rerank using BM25 algorithm for keyword-based relevance
  - rrf: Combine multiple rankings using RRF algorithm

Entity Extraction Algorithms:
  - spacy: Fast and accurate named entity recognition using spaCy's transformer models.
  - llm: Flexible entity extraction using large language models with structured JSON out
```

```
Relationship Extraction Algorithms:
  - dependency-parsing: Extract relationships using spaCy's dependency parser to identif
  - llm: Flexible relationship extraction using large language models with structured JS

Clustering Algorithms:
  - leiden: Advanced community detection using the Leiden algorithm. Improved quality ov
  - louvain: Fast community detection using the Louvain algorithm. Good for large graphs
$
```

You can also see what your current selections are.

```
$ task cli -- state show
Current State:

  Workspace: [1] BOSCH BMV080
  Session: [2] with context
$
```

4 The Implementation Story

First I started big, real big. I made a very nice looking website and got everything hooked up to the server. I got basic functionality working and tested. Things seemed fine until I wanted to be able to scale. To scale, I started using RabbitMQ, had the server post work for workers to complete. Halfway into implementing the workers, I had a realization, boy this is going to be a lot more complex than what is strictly necessary. So I trashed the workers, got rid of RabbitMQ, removed the website, turned the server into just a CLI. This sped up development by a lot as it's way easier to debug a cli application than it is a website with background workers feeding from a queue. However, at this point I was already 2 weeks into the project hahah. I should have focused on getting a MVP application out way sooner. I have a lot of tests for the application. So everything should mostly be working. I do still want to go the direction of a website with background workers in the future, but that can be for a personal project. The point of this project for this class was to showcase different rag systems and how they differ.

5 RAG

A refresher on what is RAG (Retrieval Augmented Generation): Have you ever noticed how large LLM only have context for up to a certain date? Sadly, LLMs are designed to complete the prompt, no matter what, even if it needs to make something up. Many of the hallucination issues we find with LLMs are due to a lac of context. Rag will auto-inject context into an LLM depending on the user query. But how does it know what context to give and how do we know too much context won't be given? Wouldn't it be best to give the minimum best quality context to the LLM to cut down on costs? You are absolutely correct!

This is the magic of RAG systems. We can cut down on costs but still get the best context for the users questions.

5.1 Vector RAG

I have found that Vector Rag systems are way easier to implement. I read about this fact, but now I know first hand. Vector RAG systems do a good job when questions are semantically similar to the provided documents, but if a user starts asking general questions, Vector RAG struggles. Vector RAG is a lot faster to parse documents and be ready for user queries tho.

5.2 Graph RAG

Kind of difficult to get working but man once you do its kind-of a thing of beauty. Much slower than Vector RAG, but if you have the resources and time, it's totally worth the pain. You can ask general questions, and it will give a good response this is there Vector RAG does poor. However, Graph RAG seems to struggle with semantically similar questions to the documents.

5.3 Vector RAG vs Graph RAG

Depends on what you are trying to do, it turns out. But if in doughty use Graph RAG.

5.4 Hybrid RAG

A hybrid RAG system would use both a graph and vector database, leveraging the benefits both bring to the table. However a Hybrid RAG system is very heavy on system resources, so it might not be for everyone.

6 Conclusion

A good project, learned lots while making it. Still lots of work to do, but I think the MVP is done.

6.1 Future Work

Citations

Hybrid rag

Wikipedia Injection

Multi User

Website

Kubernetes Deployment

7 References

@inproceedings{10.1145/3626772.3661370, author = {Xu, Zhentao and Cruz, Mark Jerome and Guevara, Matthew and Wang, Tie and Deshpande, Manasi and Wang, Xiaofeng and Li, Zheng}, title = {Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering}, year = {2024}, isbn = {9798400704314}, publisher = {Association for Computing Machinery}, address = {New York, NY, USA}, url = {https://doi.org/10.1145/3626772.3661370}, doi = {10.1145/3626772.3661370}, abstract = {In customer service technical support, swiftly and accurately retrieving relevant past issues is critical for efficiently resolving customer inquiries. The conventional retrieval methods in retrieval-augmented generation (RAG) for large language models (LLMs) treat a large corpus of past issue tracking tickets as plain text, ignoring the crucial intra-issue structure and inter-issue relations, which limits performance. We introduce a novel customer service question-answering method that amalgamates RAG with a knowledge graph (KG). Our method constructs a KG from historical issues for use in retrieval, retaining the intra-issue structure and inter-issue relations. During the question-answering phase, our method parses consumer queries and retrieves related sub-graphs from the KG to generate answers. This integration of a KG not only improves retrieval accuracy by preserving customer service structure information but also enhances answering quality by mitigating the effects of text segmentation. Empirical assessments on our benchmark datasets, utilizing key retrieval (MRR, Recall@K, NDCG@K) and text generation (BLEU, ROUGE, METEOR) metrics, reveal that our method outperforms the baseline by 77.6% in MRR and by 0.32 in BLEU. Our method has been deployed within LinkedIn's customer service team for approximately six months and has reduced the median per-issue resolution time by 28.6%.}, booktitle = {Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval}, pages = {2905–2909}, numpages = {5}, keywords = {knowledge graph, large language model, question answering, retrieval-augmented generation}, location = {Washington DC, USA}, series = {SIGIR '24} }

@misc{edge2025localglobalgraphrag, title={From Local to Global: A Graph RAG Approach to Query-Focused Summarization}, author={Darren Edge and Ha Trinh and Newman Cheng and Joshua Bradley and Alex Chao and Apurva Mody and Steven Truitt and Dasha Metropolitansky and Robert Osazuwa Ness and Jonathan Larson}, year={2025}, eprint={2404.16130}, archivePrefix={arXiv}, primaryClass={cs.CL}, url={https://arxiv.org/abs/2404.16130}, }

@article{Traag_2019, title={From Louvain to Leiden: guaranteeing well-connected communities}, volume={9}, ISSN={2045-2322}, url={http://dx.doi.org/10.1038/s41598-019-41695-z}, DOI={10.1038/s41598-019-41695-z}, number={1}, journal={Scientific Reports}, publisher={Springer Science and Business Media LLC}, author={Traag, V. A. and Waltman, L. and van Eck, N. J.}, year={2019}, month=mar }

@article{Pan_2024, title={Unifying Large Language Models and Knowledge Graphs: A Roadmap}, volume={36}, ISSN={2326-3865}, url={http://dx.doi.org/10.1109/TKDE.2024.3352100}, DOI={10.1109/tkde.2024.3352100}, number={7}, journal={IEEE Transactions on Knowledge and Data Engineering}, publisher={Institute of Electrical and Electronics Engineers

(IEEE)}, author={Pan, Shirui and Luo, Linhao and Wang, Yufei and Chen, Chen and Wang, Jiapu and Wu, Xindong}, year={2024}, month=jul, pages={3580–3599} }