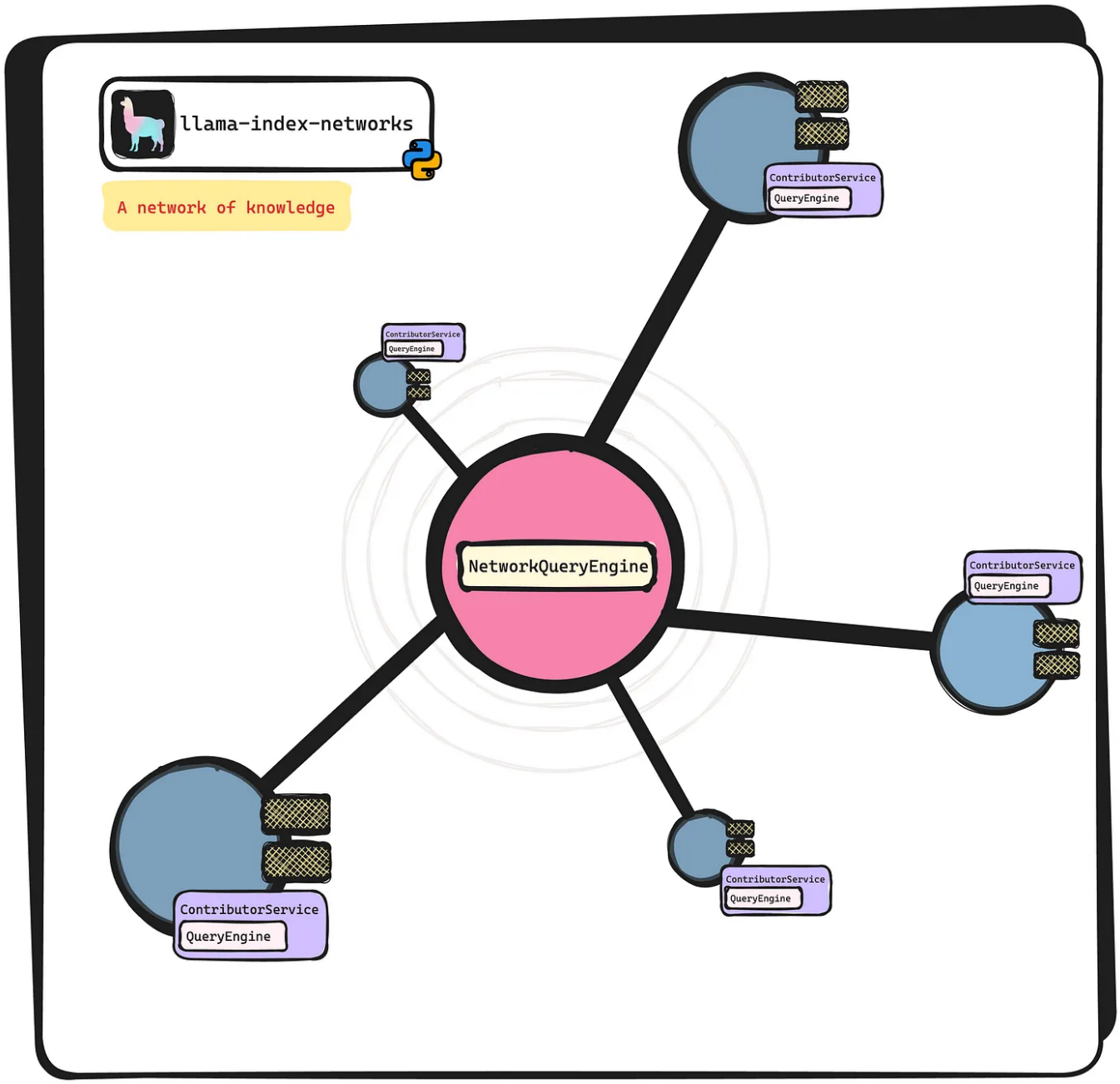
Querying a network of knowledge with llama-index-networks

Andrei · Follow

Published in LlamaIndex Blog · 6 min read · 22 hours ago



The new llama-index extension package enables rapid creations of networks of RAG systems. Data suppliers can build powerful RAGs over their data and expose it behind a ContributorService in order to join a LlamaIndex Network.

The main premise behind RAG is the injection of context (or knowledge) to the LLM in order to yield more accurate responses from it. As such, a crucial component to a RAG system is the data source from which it gets its knowledge. It's intuitive then to reason that the more knowledge that the RAG system can tap into, the better it would ultimately become (in terms of answering queries of potentially both depth and breadth). The spirit of this

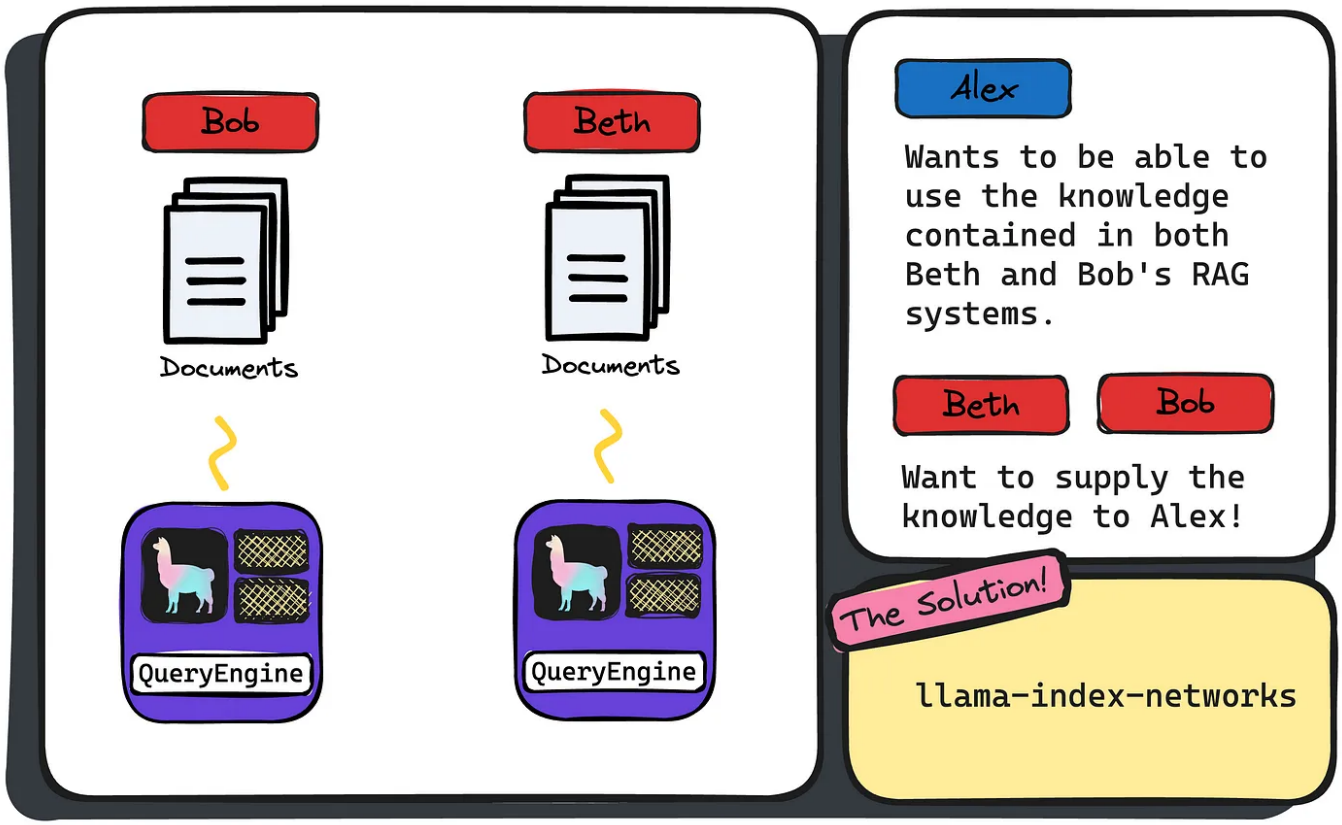
concept is not so different from that found in basically every other data-driven discipline — access to more (good) data, that is subsequently and effectively used, usually leads to better outcomes.

It is with that backdrop, that we're thrilled to announce the release of our newest latest `llama-index` library extension, called `llama-index-networks`. This library extension makes it possible to build a network of RAGs built on top of external data sources and supplied by external actors. This new network paradigm allows for a new way for data suppliers to provide their data to those who want it in order to build more knowledgeable systems!

In this blog post, we'll introduce the main classes of the new extension library and show you how in just a couple lines of code, you can make your QueryEngine ready to contribute as part of a network of RAGs. We'll also share ideas of what this can mean for how data suppliers actually supply their data to consumers within this new era of LLMs.

A note on terminology: in this post, we use `llama-index-networks` to refer to the actual extension, whereas `llama-index[networks]` refers to an installation of `llama-index` that comes with the `llama-index-networks` extension.

The story of Alex, Beth, and Bob



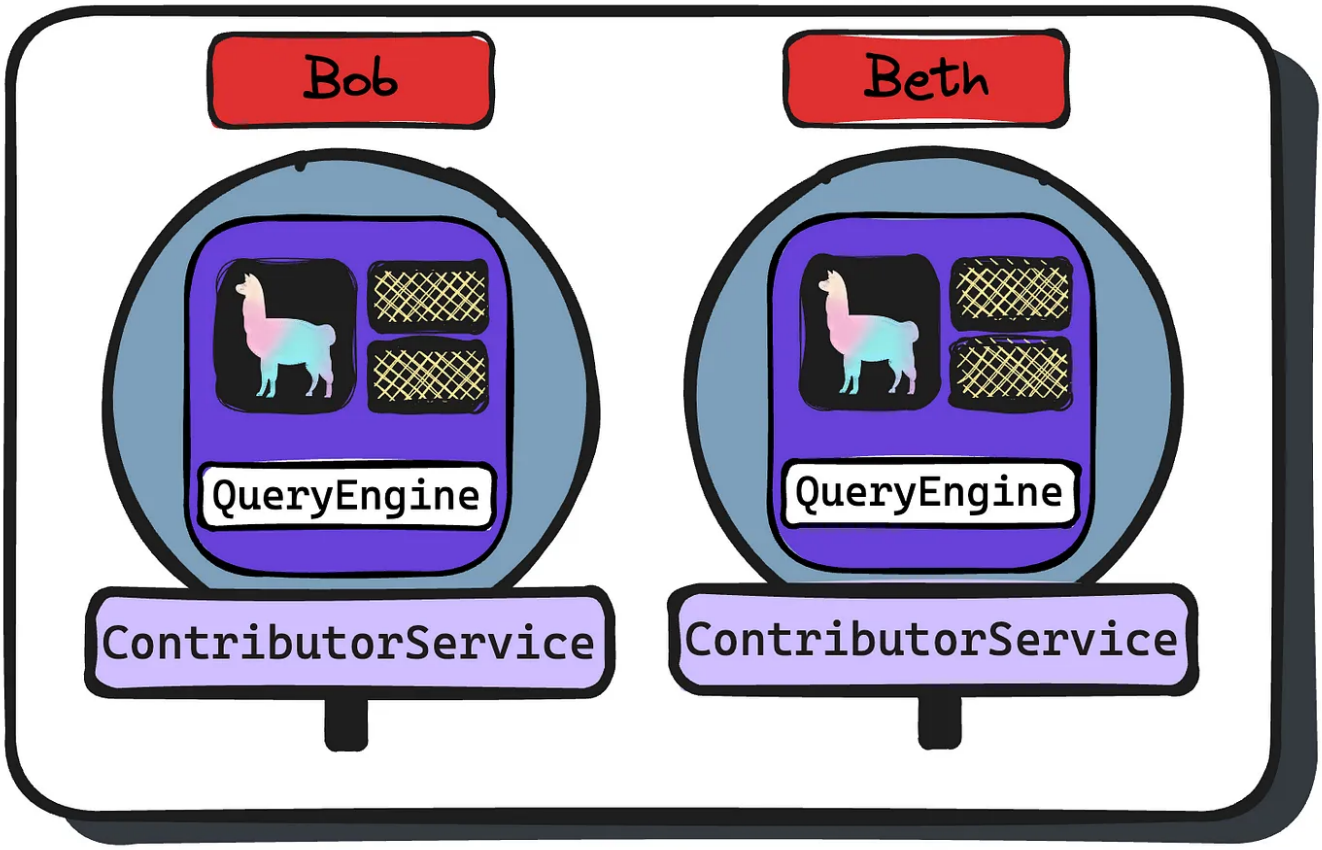
An illustrative example of actors in a network and their problem statements.

To illustrate how the llama-index-networks package can be used, we consider three made-up characters, Alex, Bob, and Beth, and the following scenario:

- Both Bob and Beth each have their own set of documents and both have already built quite the outstanding RAG systems over them (using llama-index of course!)
- Alex has heard about these insightful documents that both Bob and Beth have and would like to be able to query the individual RAGs built over the both of them.
- Bob and Beth, being as kind as they are (or, perhaps they were paid some undisclosed dollar amount), agree to give access to Alex to their RAG systems.

To facilitate this new fashion of knowledge exchange, they agree to setup a network of RAGs that Alex can query.

Bob and Beth build a web service over their RAGs



ContributorService is built around a QueryEngine.

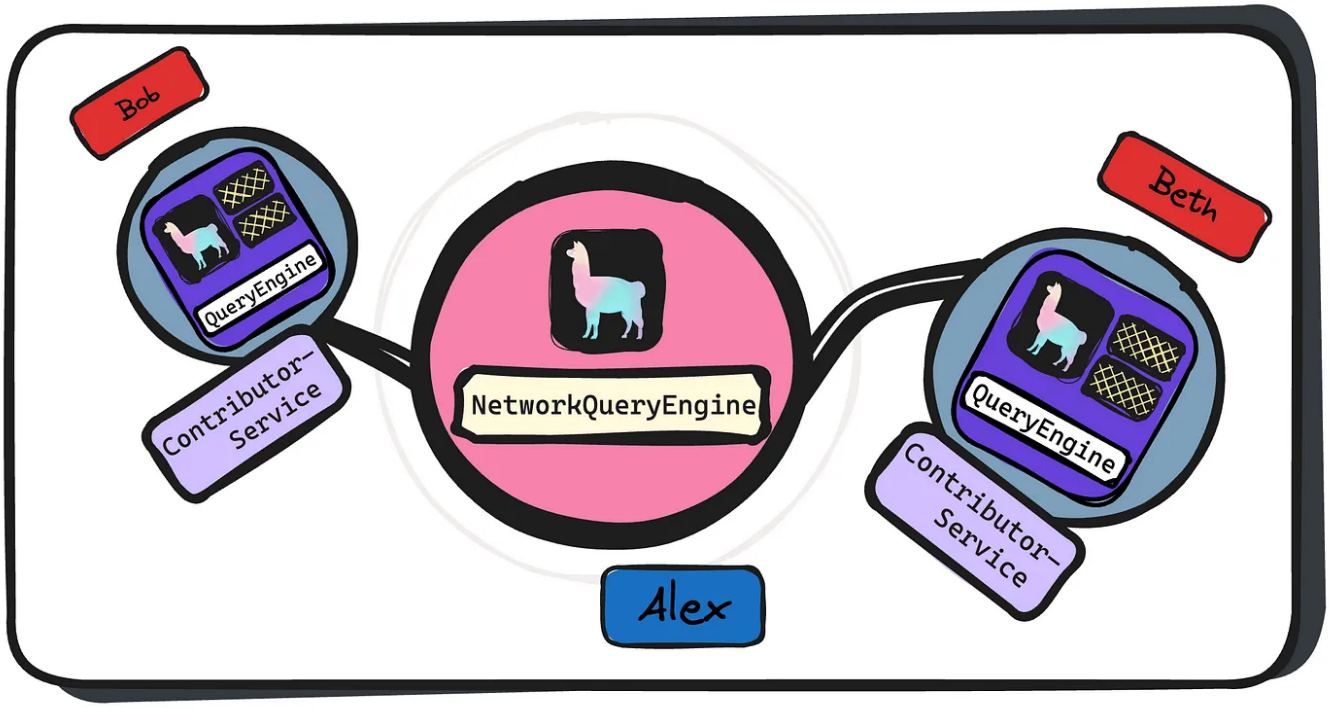
With the `llama-index-networks` package, Bob and Beth can make their respective `QueryEngine`'s ready to participate in the network with only a few lines of code.

```
"""Beth's contributor service file.

Beth builds her QueryEngine and exposes it behind the standard
LlamaIndex Network Contributor Service.

NOTE: Bob would probably make use of Docker and cloud
```

```
compute services to make this production grade.
"""

from llama_index.networks.contributor import ContributorService
import uvicorn

beth_query_engine = ...
beth_contributor_service = ContributorService.from_config_file(
    ".env.contributor",  # settings/secrets for the service
    beth_query_engine
)


if __name__ == "__main__:
    uvicorn.run(beth_contributor_service.app, port=8000)
```

Bob would use similar lines of code to make his `QueryEngine` ready to contribute to any LlamaIndex network. Note, that the dotenv file `.env.contributor` contains settings for the service as well as any necessary api keys (e.g., `OPENAI_API_KEY` or `ANTHROPIC_API_KEY`), which under the hood is implemented as REST service using FastAPI.

Alex builds a NetworkQueryEngine



Alex builds a NetworkQueryEngine that connects to Beth and Bob's individual ContributorService's.

For Alex's part, he uses the `NetworkQueryEngine` class of the `llama-index-networks` extension to be able to connect to both Beth and Bob's `ContributorService`'s.

```
"""Alex's network query engine.

Alex builds a NetworkQueryEngine to connect to a
list of ContributorService's.
"""

from llama_index.networks.contributor import ContributorClient
from llama_index.networks.query_engine import NetworkQueryEngine
from llama_index.llms.groq import Groq

# Use ContributorClient to connect to a ContributorService
beth_client = ContributorClient.from_config_file(
    env_file=".env.beth_contributor.client"
)
bob_client = ContributorClient.from_config_file(
    env_file=".env.bob_contributor.client"
)
contributors = [beth_client, bob_client]
```

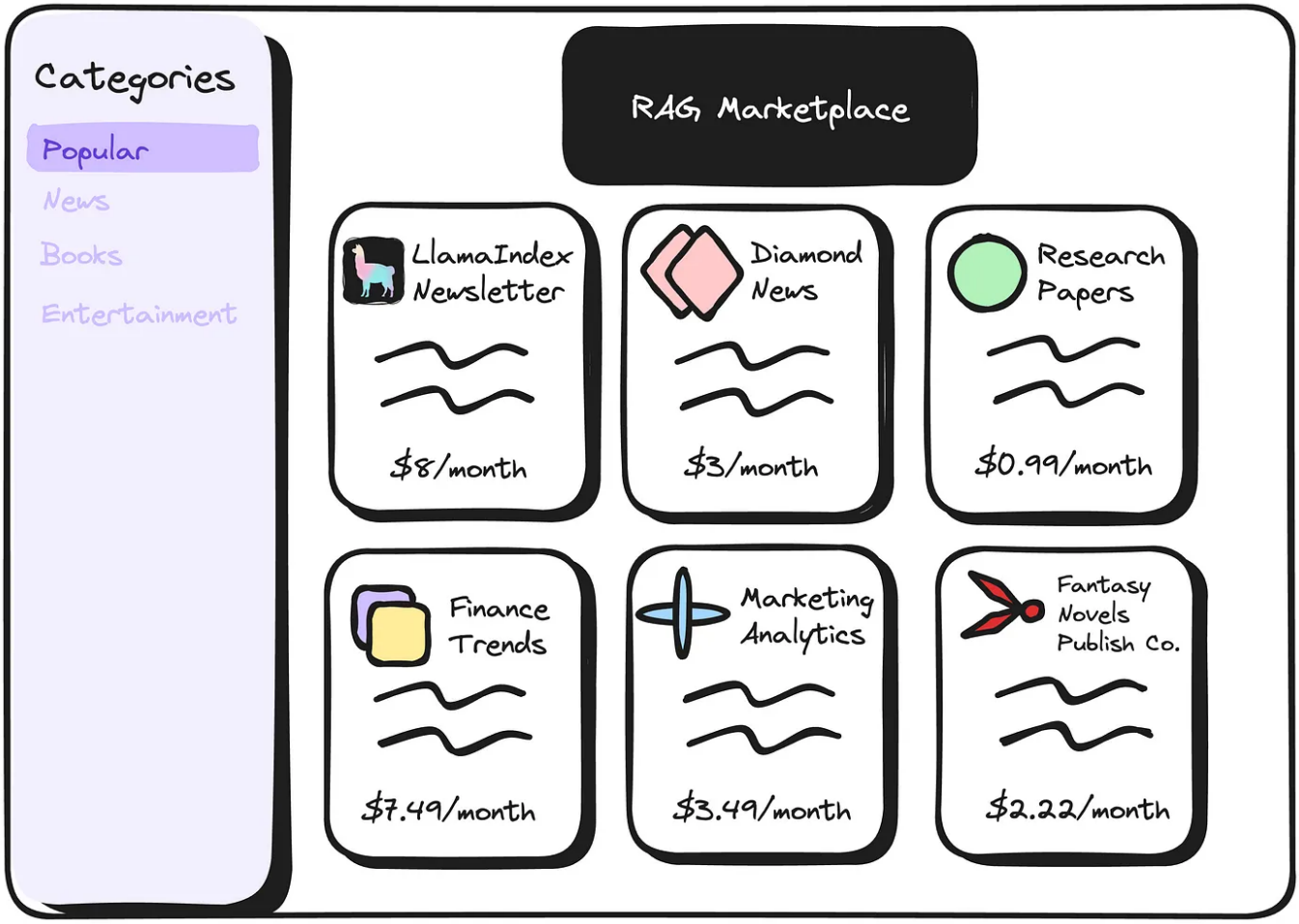
```
# NetworkQueryEngine
llm = Groq()
network_query_engine = NetworkQueryEngine.from_args(
    contributors=contributors,
    llm=llm
)

# Can query it like any other query engine
network_query_engine.query("Why is the sky blue?")
```

Here the dotenv files store the service parameters such as an `api_url` required to connect to the `ContributorService`.

Before moving on to the next section of this blog post, we'll take the next few lines to explain a bit of what's involved under the hood when Alex query's his `NetworkQueryEngine`. When the `query()` method is invoked (Async is also supported!), the query is sent to all contributors. Their responses are stored as new `Nodes` and a response is synthesized (with the associated `ResponseSynthesizer`). After reading that, some may notice that this is the usual pattern when working with our `QueryEngine` abstractions; and, that indeed was the point. Using a `NetworkQueryEngine` should be very similar to how you would use any other `QueryEngine` in our library!

This marks the end of our little story about Alex, Bob and Beth. Before wrapping up this blog post, we first provide a few potential opportunities that may arise through the usage of `llama-index-networks`.

A new world of data supply and consumption



RAG marketplaces is a use case that can be made possible with llama-index[networks].

One possible world that could easily be powered by `llama-index[networks]` are marketplaces for RAG. A place where data suppliers package their data in the form of RAGs to data consumers look to expand their own query system's knowledge. Potential RAG (data) suppliers could be your local newspaper, book publishing companies, etc.

In this new data supply and consumption model, there is more onus on the data supplier to prepare the data in a fashion that makes it easier to consume — specifically, the suppliers own the building of the query engine. This should benefit the data consumer greatly since, with a standard interface that is provided by the likes of a `ContributorService` (that encapsulates a `QueryEngine`), they can get access to the knowledge they seek from the data

easier than ever before (i.e., in relation to traditional data marketplaces that exchange raw data).

It is with this kind of vision that we've built `llama-index[networks]` to make it: (i) easier for data suppliers to supply the knowledge contained in their data in new and arguably more effective ways, and (ii) easier for data consumers to connect to these new forms of external knowledge.

Internal networks: another potential use case

In addition to powering RAG marketplaces, we foresee the need of connecting RAGs that an overarching company may own, but not necessarily manage. More concretely, a franchise may have the rights to the data across all of its operations. And, while they could build a "central", monolithic RAG over the entire database, it may still be more efficient and effective to build RAGs over the individual operators and query these instead.

The idea of exchanging information to build better, more knowledgeable systems is not new. However, the idea of using RAGs to facilitate that exchange may be (to our knowledge, it is), and we believe that both existing and new use cases requiring such collaboration can benefit from concept.

A quick note on privacy

An important consideration in the collaboration of data is privacy and security. It bears mentioning that the examples above assume that the data that is being shared across the network is in compliance with data privacy and security laws and regulations. It is our belief that as this technology grows, that the necessary features and capabilities will be developed and incorporated to facilitate in-compliance RAG networks.

Check out the demo to learn more!

To see an actual demo of a network connecting to a set of contributors, check out the Github repository code for `llama-index-networks` and navigate to the `examples/demo` subfolder.

Llm | Retrieval Augmented | AI | Distributed | Machine Learning

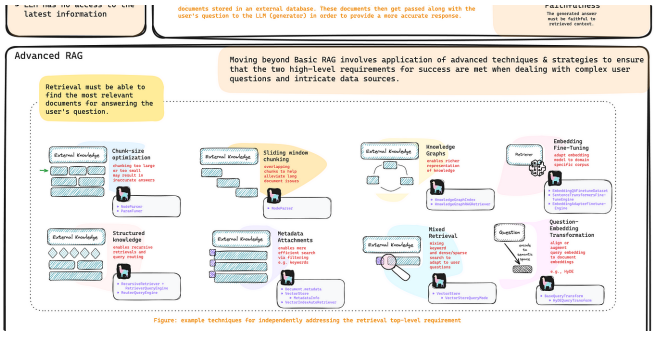
Written by Andrei

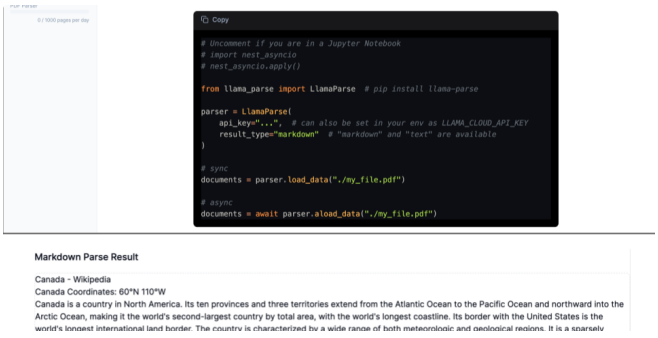
Follow

467 Followers · Writer for LlamaIndex Blog

Founding software/machine-learning engineer at LlamaIndex. (https://ca.linkedin.com/in/nerdai)

More from Andrei and LlamaIndex Blog





Andrei in LlamaIndex Blog

A Cheat Sheet and Some Recipes For Building Advanced RAG

It's the start of a new year and perhaps you're looking to break into the RAG scene by...

7 min read · Jan 5, 2024

1.5K 4

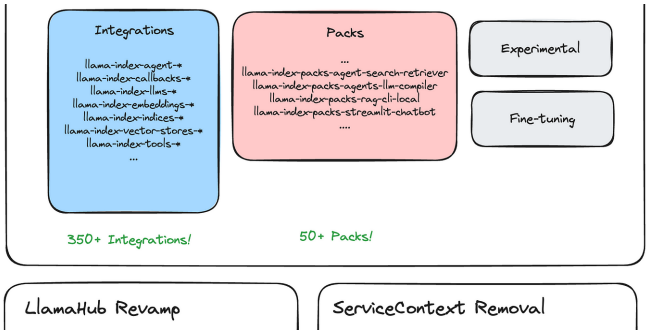
Jerry Liu in LlamaIndex Blog

Introducing LlamaCloud and LlamaParse

Today is a big day for the LlamaIndex ecosystem: we are announcing LlamaCloud, ...

8 min read · Feb 20, 2024

1K 11



LlamaIndex in LlamaIndex Blog

LlamaIndex v0.10

Today we're excited to launch LlamaIndex v0.10.0. It is by far the biggest update to our...

9 min read · Feb 12, 2024

438 7

Andrei in integrateai-skunkworks-lab

ToF: Decentralization of AI for Web3

One of the primary concepts to web3 is decentralization. In this context,...

6 min read · Mar 16, 2022

71

See all from Andrei

See all from LlamaIndex Blog

Recommended from Medium

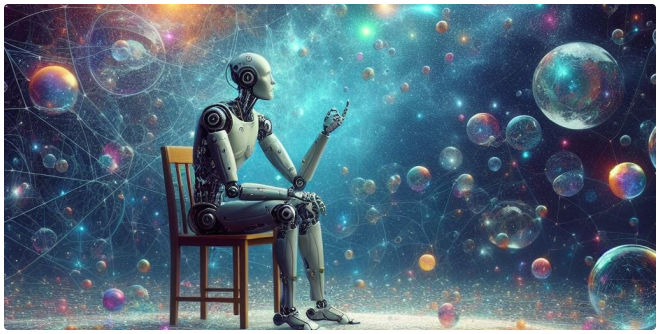
Wenqi Glantz in Towards Data Science

NeMo Guardrails, the Ultimate Open-Source LLM Security Toolkit

Exploring NeMo Guardrails' practical use cases

✦ · 13 min read · Feb 9, 2024

563



Aniket Hingane | Day Manager, Night Coder

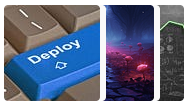
LLM Apps : Why Knowledge Graphs are super critical to know ...

AI Apps Need RAG, RAG Need Knowledge Graph ! Next part we will code RAG on KG.

✦ · 5 min read · Feb 21, 2024

58 1

Lists



Predictive Modeling w/ Python

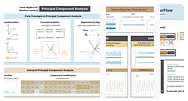
20 stories · 948 saves

Natural Language Processing

1232 stories · 718 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 318 saves



Practical Guides to Machine Learning

10 stories · 1118 saves

Khoa Le, Ph.D.

Finetuning LLM with QLoRa on financial data for sentiment...

Continuing from my previous tutorial on using RAG to enhance LLM models with more...

6 min read · Jan 13, 2024

60

Anthony Alcaraz in The Modern Scientist

Unlocking the Power of Documents with Structure-Aware AI

The Untapped Wealth of Documents

· 7 min read · 5 days ago

172 5

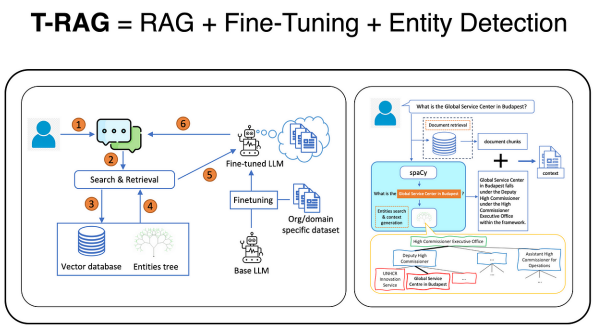
Chetankumar Khadke

Information extraction with Mistral 8×7B

Extraction of Key entities from documents using Mistral 8*7B.

14 min read · Feb 20, 2024

120 1



Cobus Greyling

T-RAG = RAG + Fine-Tuning + Entity Detection

The T-RAG approach is premised on combining RAG architecture with an open-...

5 min read · Feb 15, 2024

794 8

See more recommendations