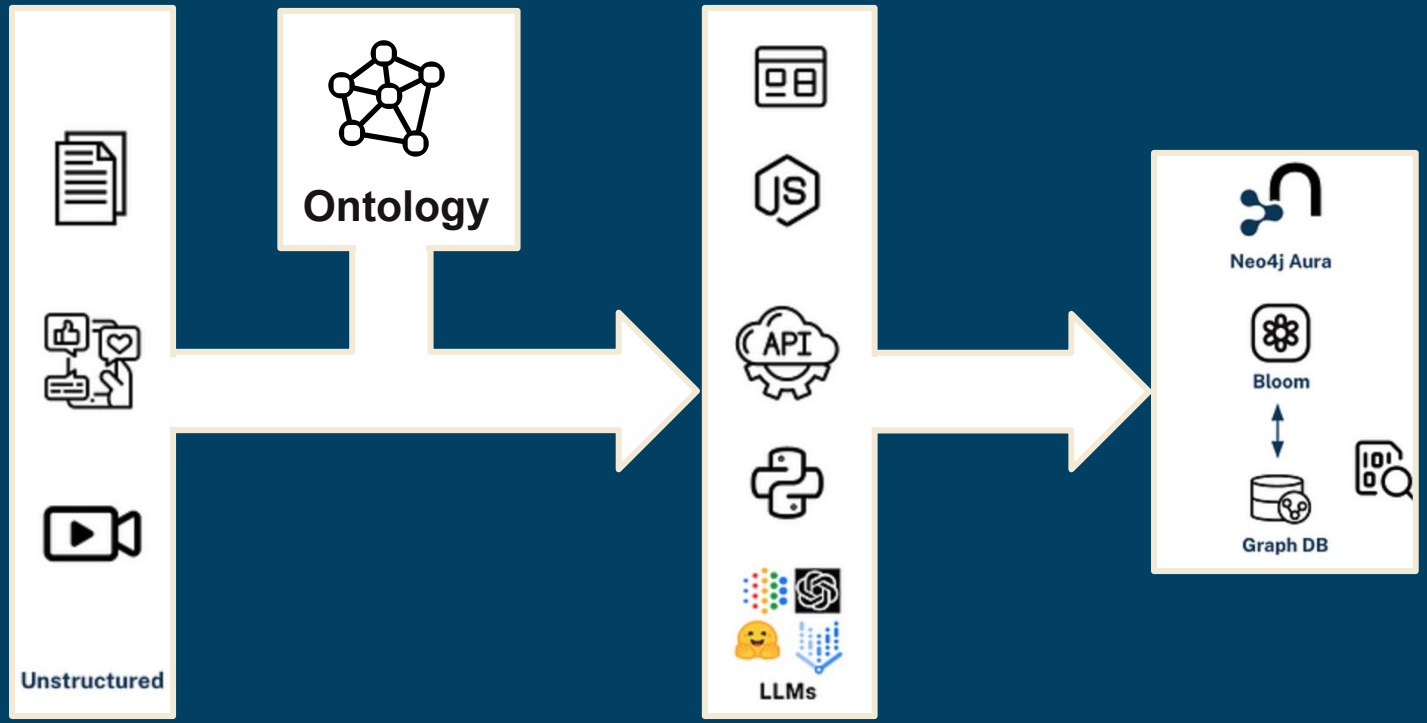
neo4j

Going Meta

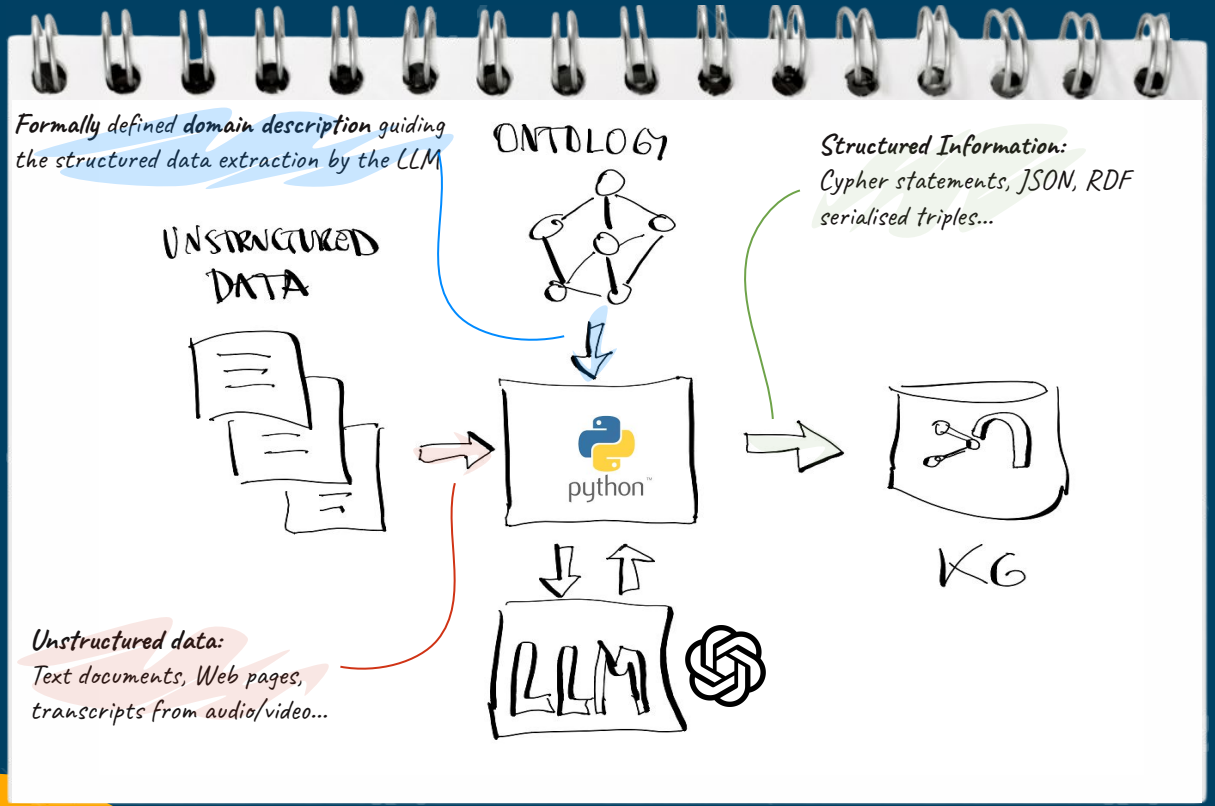
S2 - Episode 3

3 Blueprints for KG Construction from Unstructured Data

Refresher: What does it mean?



Today's approach



The Dataset

GraphRAG in Action: From Commercial Contracts to a Dynamic Q&A Agent

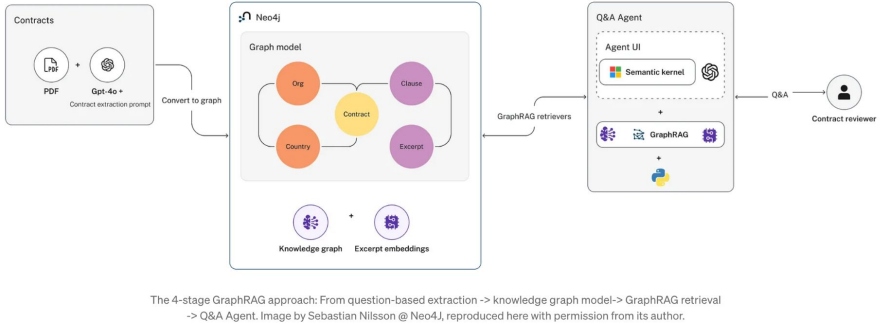
A question-based extraction approach

Ed Sandoval · Follow

Published in Towards Data Science · 23 min read · Nov 4, 2024

293

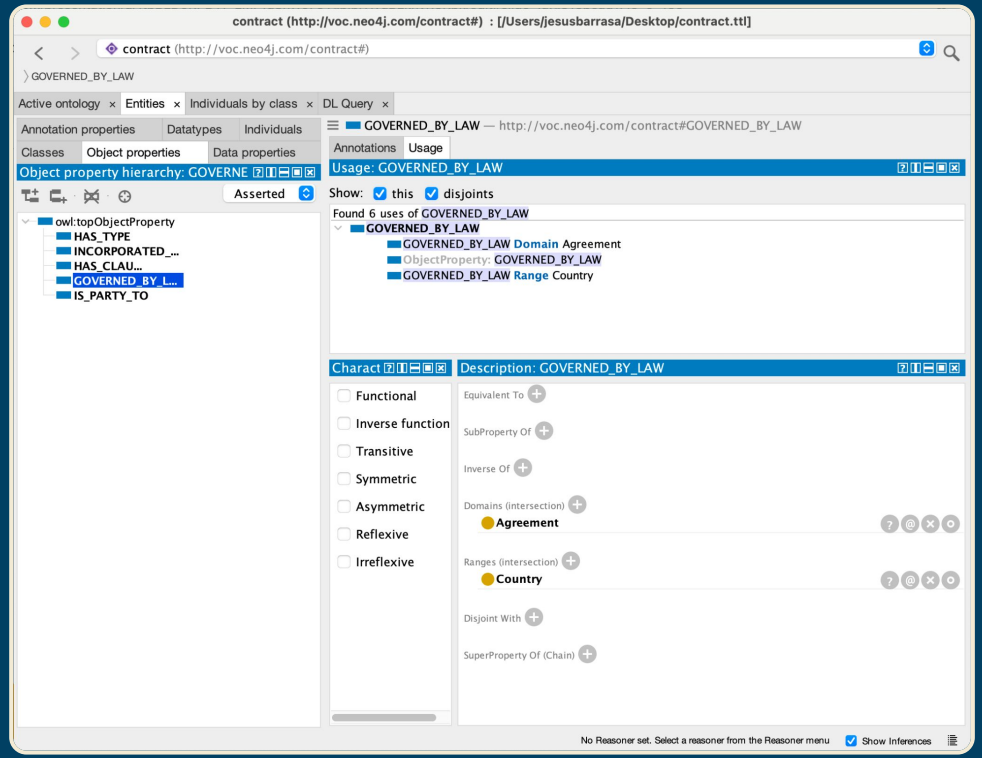
In this blog post, we introduce an approach that leverages a Gra
Augmented Generation (GraphRAG) method — to streamline the process of ingesting commercial contract data and building a Q&A Agent.



The 4-stage GraphRAG approach: From question-based extraction -> knowledge graph model-> GraphRAG retrieval -> Q&A Agent. Image by Sebastian Nilsson @ Neo4J, reproduced here with permission from its author.

https://towardsdatascience.com/graphrag-in-action-from-commercial-contracts-to-a-dynamic-q-a-agent-7d4a6caa6eb5

The Ontology



Option 1: Code Generation

APPROACH

LLM runs entity extraction informed by ontology and produces code directly executable in the DB.

Option 1: Code Generation

PROS

- Easy. Full delegation to LLM in one simple call

CONS

- Target Platform dependent
- Data + import logic mixed
- Harder to debug
- Not great version control / diff
- Guardrails post write (in Database)

Question:
Does it help to talk RDF to the LLM?

Option 2: RDF Generation + Import

APPROACH

LLM runs entity extraction informed by ontology and produces an RDF serialisation of the results.
The RDF data is subsequently imported into the Graph DB with a generic RDF importer.

Option 2: RDF Generation + Import

PROS

- Easy. Full delegation to LLM in one simple call
- The output is a Graph -> no import logic required
- Guardrails on RDF serialisation (pre or post write)
- Good version control / diff

CONS

- RDF import capability
- Less user friendly (?)

Option 3: JSON generation + Import

APPROACH

LLM runs entity extraction informed by ontology and produces a JSON representation of the results. The JSON data is subsequently imported into the Graph DB with a custom *(or generic?)* script.

Option 3: JSON generation + Import

PROS

- Guardrails can be built into entity extraction
- Efficient DB import
- Good version control / diff
- User friendly, well known format, good tooling... (JSON)

CONS

- Not all LLMs support tools/functions to define output format
- Additional artifacts: pydantic validator, import script... BUT!

Let’s try it out!

