ontotext

This is part of Ontotext's AI-in-Action initiative aimed at enabling data scientists and engineers to benefit from the AI capabilities of our products.

In previous publications, we introduced the Ontotext Knowledge Graph (OTKG) and described how it boosts our marketing team's efforts. One of the main components of OTKG is the natural language querying (NLQ) interface, which implements the Graph Retrieval Augmented Generation (Graph RAG) pattern. As discussed before, there are several different patterns for implementing a Graph RAG. In a previous post, we provided examples of how all these approaches can be implemented with Ontotext GraphDB and Ontotext Metadata Studio and what the principle differences are regarding explainability, hallucination, grounding context visibility, and other aspects.

In this sequel of the series, we're sharing insights on the different varieties we've explored for the OTKG Chat.

Graph RAG Patterns

Let's quickly recap the different Graph RAG patterns:

- **Graph as a Content Store**: This approach uses the graph as a document store. The

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

documents are divided into chunks, which are indexed in a vector database. At query time, relevant document chunks are retrieved from the graph and the LLM is prompted to answer the user's question using them.

- **Graph as an Expert or a Vocabulary**: This approach uses information modeled in the graph to aid the RAG process. A sub-graph, describing the concepts relevant to the user's question, is extracted and provided to the LLM as a "semantic context". This can be done either prior to indexing or at query time.
- **Graph as a Database**: This approach maps the user's question to a structured database query, which is executed, and the LLM is prompted to summarize the results and answer the question.

Given the nature of the users' questions, the domain, the information in the knowledge graph, and the system requirements, different RAG patterns may be appropriate. However, all present us with specific challenges and have certain limitations.
One of them has become notorious as "the NASA problem" within our team. Namely, it labels the case of our initial implementation not returning NASA as a client of Ontotext when the system is posed with the question "What US government agencies has Ontotext worked with?".
The answer from our initial implementation of the Ontotext Chat was:

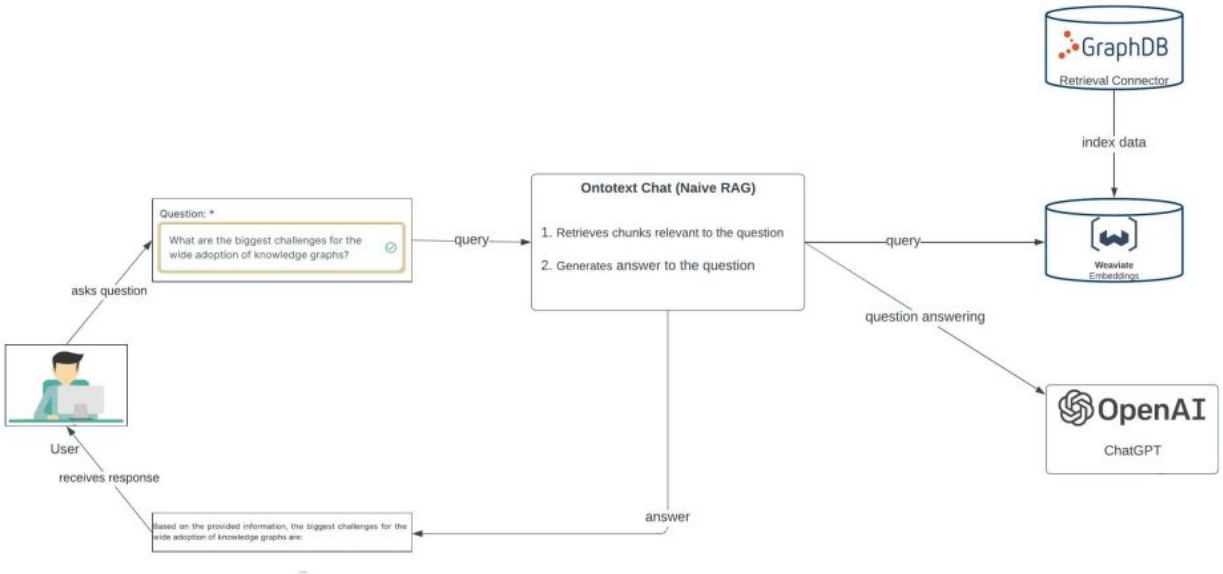
> Ontotext has worked with the US Department of Defense and US Medicare [1].

The basic problem is that none of the chunks mentioning NASA provide the context that it is a US government agency. This is not the only problem though, which makes it a good case to illustrate the limitations of the different approaches.

Graph as a Content Store

The initial implementation of the OTKG Chat is a classical RAG approach, also referred to as *Naive RAG*, which uses the graph as a Content Store and follows the traditional process of indexing, retrieval, and generation. In short, a user input is used to query relevant documents, which are then combined with a prompt and passed to the model to generate a final response.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com



We index the OTKG data in Weaviate using GraphDB's ChatGPT Retrieval Plugin Connector. We use non-overlapping chunks of equal size of 200 tokens.
For the retrieval step, we use the built-in hybrid search, which combines the results of a vector search and a keyword search (BM25F) based on the query string. The alpha parameter of the hybrid search controls how much each search affects the results. An alpha of 0 is a pure keyword search. An alpha of 1 is a pure vector search. We use 0.5 for alpha, and we fetch the first top 10 chunks.
Retrieving more chunks slows down the response time, which currently on average is around 2 seconds per question. We consider this suboptimal for use in real-life production environments. Thus, it is critical for us to be able to retrieve answers with minimal delay.
One of the Naive RAG limitations is the low recall (failure to retrieve all relevant chunks). So, in fact, the so-called "NASA-problem" is the well-known low recall limitation. To tackle this problem, we decided to explore the other Graph RAG variations.

Graph as an Expert

In order to implement the concept of using the graph as an Expert or a Vocabulary, we enrich the texts and the embeddings of the chunks by appending the descriptions of the Wikidata

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

entities at index time. We use CEEL, our Wikidata entity linking offering, to extract the Wikidata entities from the text chunks. For each of the unique Person, Location and Organization concepts mentioned in the chunks, we append their descriptions at the end. We use SPARQL queries to retrieve data from Wikidata and populate the following template that we use for the entities' descriptions:

```
{name}; an instance of {classes}; {description}; located in
{countries}; subsidiary of {parent_organizations}; owned by
{owned_by}; operates in the following industries: {industries}
```

Depending on the entity type and the data, we might omit some parts of the template. For example, the NASA description is:

```
National Aeronautics and Space Administration; an instance of
independent agency of the United States government, space agency;
American space and aeronautics agency; located in United States of
America; subsidiary of federal government of the United States;
operates in the following industries: public sector
```

So instead of this chunk:

```
Check out our demo to understand the various Industry 4.0 standards
modeled in a knowledge graph!   Knowledge graphs in the Aerospace
Industry Understandably, knowledge graphs are becoming a crucial
technology in the aerospace industry as well. While these companies
employ many standard manufacturing processes and operations, aerospace
parts manufacturing also requires advanced quality standards and
significantly more administrative effort.  Here again knowledge graphs
organize and link large amounts of data on aircraft design,
manufacturing, maintenance and performance. By linking this data, they
facilitate tasks like asset management, predictive maintenance,
documentation management, mission planning, risk management, aircraft
design and optimization, and anomaly detection.  Since 2020, Ontotext
has been working with NASA and, interestingly, six out of the top ten
aerospace companies in the world use GraphDB in some part of their
```

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

```
operations. Logistics &amp;amp; Supply Chain Two other industries where
knowledge graphs are hitting the mark are logistics and supply chain
management.
```

we end up with this enriched chunk where the original text is appended as follows:

```
<original chunk text>
Ontotext; an instance of software company, company; global leader in
enterprise knowledge graph technology and semantic database engines;
located in Bulgaria; subsidiary of Sirma Group Holding; operates in
the following industries: knowledge graph, Semantic Web, software
industry, graph database
National Aeronautics and Space Administration; an instance of
independent agency of the United States government, space agency;
American space and aeronautics agency; located in United States of
America; subsidiary of federal government of the United States;
operates in the following industries: public sector
```

Experiments

We have 4 chunks mentioning NASA as a customer of Ontotext. The goal of the experiments is to check if one of these chunks will be ranked among the first 10. Initially, since the first 10 chunks don't include any mention of NASA, we have to increase the limit. We have to keep in mind that changing the limit affects the results, meaning it's not the same, if we fetch 100 chunks from Weaviate and get the first 10 compared to if we fetch only 10 chunks from Weaviate.
So, for the purposes of the experiments, we increase and fix the limit to 3100. This limit allows us to fetch all 4 chunks mentioning NASA, if we use alpha=0.5 and alpha=1 and the 3 chunks that can be fetched with alpha=0 (even if we increase the limit with alpha=0, not all 4 chunks can be retrieved).

Baseline / Graph as a Content Store

In the table below, we see the results with different alpha values for the 4 chunks. We highlight the minimum value on each row in bold.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

alpha / chunk id	39387_1	46199_1	49577_3	48756_1
0.5 (hybrid)	**1672**	1682	1938	2170
1 (vector search)	**1302**	1321	2023	1735
0 (keyword search, BM25)	3013	3008	**1505**	-

We use alpha = 0.5 and the first chunk mentioning NASA is with rank 1672. If we use only vector search or only keyword search, the results will improve a bit, but still quite far from the first top 10 results.
It is worth noting that in this case, the content is indexed in Weaviate for approximately 3 minutes.

Graph as an Expert

The content with this approach is indexed in Weaviate for approximately 1 hour and 25 minutes. This time can be reduced substantially via optimizations, but there will always be an overhead related to fetching context and augmenting chunks. These are the results:

alpha / chunk id	39387_1	46199_1	49577_3	48756_1
0.5 (hybrid)	**107**	111	162	1145
1 (vector search)	**383**	406	1041	1581
0 (keyword search, BM25)	164	163	**50**	362

Here, we see a huge improvement for all possible values of alpha, but still we are far from the target of 10.
Next, we tried the query rewrite technique to enrich the user question using keywords generated by GPT. The prompt we used is:

```
Give thirty words that may logically be in answer to this question: {}
```

So, for the user question “What US government agencies has Ontotext work with?” we get:

```
What US government agencies has Ontotext worked with? CIA, FBI, NSA,
NASA, DHS, DOD, DOE, NIH, CDC, EPA, USDA, NIST, NOAA, USGS, DARPA,
IARPA, DOT, DOJ, VA, GSA
```

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

Using this query rewrite we get the following results:

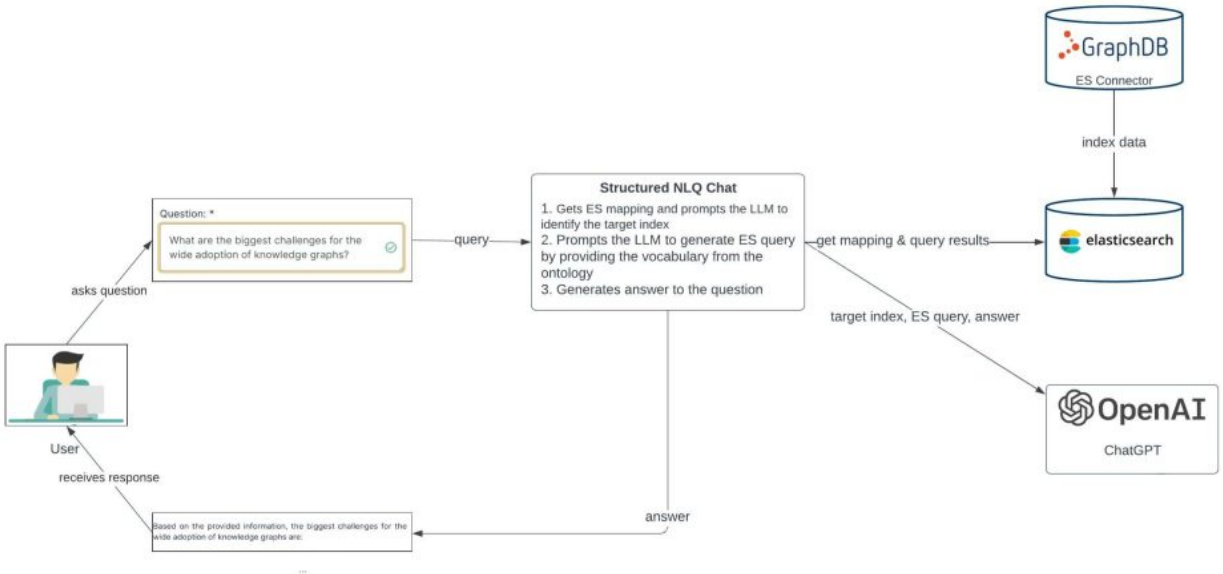
alpha / chunk id	39387_1	46199_1	49577_3	48756_1
0.5 (hybrid)	**16**	18	19	186
1 (vector search)	**151**	169	578	1216
0 (keyword search, BM25)	22	23	**12**	50

These results look better, but still if we want to stick to the hybrid search, the top chunk mentioning NASA comes as the 16th candidate. The aftermath is that even when we use "graph as an Expert" to add context, we hit another problem. One of the fundamental limitations of the "graph as a Content Store" can be summarized as follows:
The *Chunky RAG Axiom*: *If the information required to answer the question correctly is spread across more chunks than we retrieve, the answer may be correct, but incomplete.*
For example, we can hardly answer questions like "How many customers does Ontotext have?", "Which is the latest release of GraphDB?", "Which is the oldest and the newest Research Projects of Ontotext, and what was the role of the company in the project?", or other types of questions that might need some sort of aggregation.

Graph as a Database

To address these limitations, we implemented a new Chat with graph as a Database. The diagram below shows what is the RAG architecture.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com



It's quite similar to the Naive RAG Chat and to what we presented in our blog post about the integration of GraphDB in LangChain, but instead of prompting the LLM to generate SPARQL queries, we generate Elasticsearch (ES) queries. Also, instead of indexing the content into Weaviate by splitting into chunks and indexing their texts and embeddings, we use the GraphDB Elasticsearch Connector to index the OTKG data into ES.

Implementation

We have several ES indices: content, people, organizations, and products. The "organizations" index contains some clients and partners of Ontotext, only those, which appear on our website under Partners and Customers, or which were part of our organized events. So, for now, this is not an exhaustive list (and never will be due to NDA agreements). Whether to generate SPARQL or ES queries depends on the types of users' questions and the ontology. Both approaches have pros and cons. The pros of the SPARQL approach is that when sometimes the LLM generates a query with missing prefixes, syntactic or other errors, and we try to amend this by prompting the LLM to correct it, this works pretty well. For ES the LLM can hardly correct the query based on the error message, especially when the query contains aggregations on text fields (fielddata is disabled by default on text fields in ES).

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

An advantage of the ES queries approach is that it's pretty straightforward to pass a subset of the taxonomy, which reduces the prompt size. We first ask the LLM to identify the target ES index and then provide a subset of the taxonomy, which is relevant for this specific index. It's not obvious how to do this for SPARQL, and it's not yet explored.
Let's now check what the answer from this chat is.

> Ontotext has worked with the following US government agencies: Food and Agriculture Organization and NASA.

Pretty nice! The LLM generates a correct query and retrieves all relevant results. The peculiarity that the UN Food and Agriculture Organization (FAO) turns up as an US government agency is related to limitations of the underlying OTKG data model, where we can only ask for "organization from the public/government sectors, which is headquartered in the USA".

For the nerds out there, here is the generated ES query:

```
{
    "size": 5,
    "query": {
        "bool": {
            "must": [
                {
                    "term": {
                        "country":
"https://kg.ontotext.com/resource/country/united_states_of_america"
                    }
                },
                {
                    "term": {
                        "type":
"https://kg.ontotext.com/resource/orgType/government"
                    }
```

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

```
                }
            ]
        }
    },
    "_source": [
        "name"
    ]
}
```

However, one may notice that now we get NASA, but since the US Department of Defense and US Medicare are not added to the ontology, we miss those. To combine both, we can use Modular RAG and provide to the LLM both structured and unstructured data from the graph. This is what we are currently working on, so stay tuned for our next post in this series.

To wrap it up

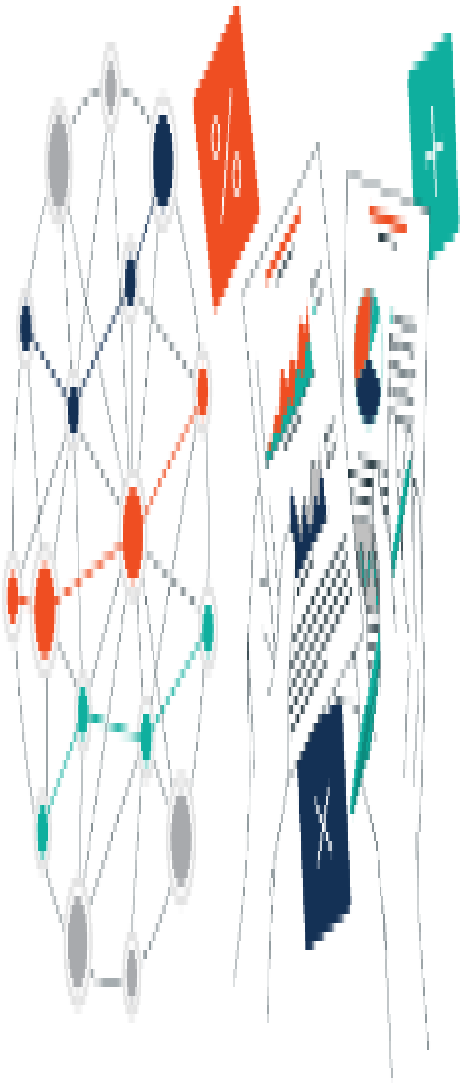
We reviewed different Graph RAG approaches and a few techniques to overcome some limitations of the Naive RAG. We used our state-of-the-art entity linking model CEEL to augment document chunks with context extracted from Wikidata. Using a public knowledge graph like this can massively reduce the time and effort for building proprietary resources to serve the same purpose. We also experimented to use LLMs for NLQ generation.
Both approaches show that using knowledge graphs improves the quality of the RAG. However, depending on the type of users' questions we aim to answer, we need to combine structured and unstructured data from the graph to achieve better quality.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

Does this resemble your use case?

Book a Discovery Call



Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com