Text extraction from Knowledge Graphs in the Oil and Gas Industry

Laura P. Navarro[12], **Elvis A. de Souza**[23], **Marco A. C. Pacheco** [12]

[1] Department of Electrical Engineering (ELE/PUC-Rio)
[2]Applied Computational Intelligence Lab. (ICA/PUC-Rio)
[3]Institute of Mathematical and Computer Sciences (ICMC/USP)

laurap@aluno.puc-rio.br, elvis.desouza99@gmail.com

Abstract. *This paper presents a detailed methodology for extracting and analyzing data from a knowledge graph designed to store complex geological information. Our pipeline was designed after a deep understanding of the KG, focuses on browsing, querying and transforming data using curated text templates. The extraction methodology is based on graph triples, key classes, properties and relationships, which ensures the relevance and truthfulness of the data obtained. With the recent advancements in neural large language models, which perform exceptionally well on open-domain tasks, our work addresses the challenge of presenting LLMs with accurate closed-domain data—originating from graph-based sources—in a readable and accessible textual format.*

1. Introduction

Efficient data extraction from knowledge graphs is fundamental for research and analysis across several fields [Ali et al. 2022]. Generating textual information from knowledge graphs involves identifying and extracting relevant information from the dataset, and may include collecting facts, relationships, and entities. Once this information has been extracted using SPARQL queries, it can be utilized to create coherent narratives or descriptions based on the extracted data [Ribeiro et al. 2020, Mizell et al. 2014, Koncel-Kedziorski et al. 2019].

Our goal is to generate accurate textual data to be ingested by a domain-specific Q&A system based on PetroKGraph, a knowledge graph from the oil and gas domain. Besides textual generation, we also generate questions and answers to evaluate the performance of our closed-domain Q&A system. In this context, the correct interpretation of data from a knowledge graph is crucial to understand the distribution and characteristics of geological formations and wells. Based on a small knowledge graph with 4,472 entities (and classes and relations) and using 29 text templates, we generate a dataset with 1,414 simple and 8,805 complex questions, along with 10,219 answers and respective contexts.

2. Related Work

Authors in [Peng et al. 2017] create a subgraph using keywords and then augment it using terms from SPARQL queries.[1] [Elbassuoni et al. 2010] search the knowledge graph

[1]SPARQL [Harris et al. 2013] is a standard query language for finding and processing data in graph databases, it enables users to execute complicated queries to retrieve particular information. The main types of queries include SELECT, ASK and DESCRIBE to extract data by matching graph patterns.

dominated by extended SPARQL, which augments the triple patterns in SPARQL with keyword criteria. [Koncel-Kedziorski et al. 2019] explore and improve the process of generating multi-sentence text from automatically extracted information using knowledge graphs; in this work they introduce, GraphWriter, a model which leverages the structured representation of KG form to enhance the quality and coherence of generated texts.

Other approaches, such as graph-to-text generation, aim to produce fluent texts from graph-based data [Colas et al. 2022, Yuan and Färber 2023]. In order to assess the effects of various task-adaptive pretraining techniques in graph-to-text generation, [Ribeiro et al. 2020] examine pretrained language models, BART [Lewis 2019] and T5 [Raffel et al. 2020]. For our approach, we use text templates tailored by domain specialists, therefore the use of generative language models was not required.

3. PetroKGraph

Knowledge Graphs (KG) include various types of vertices and edges, which may be denoted as $G = (V, E, R)$. Here, V is a collection of vertices or entities, R is a collection of relationships, and each edge is specified by its relation type $r \in R$, which means that edges can be represented as triplets. RDF (Resource Description Framework) triples follow the structure of subject-predicate-object to represent data [Manola et al. 2004].

[Cordeiro et al. 2024b] present a methodology for extracting geoscientific entities and relations from technical documents, which was used to populate a knowledge graph called PetroKGraph, specifically developed for the oil and gas industry. The construction of PetroKGraph involved leveraging natural language processing (NLP) resources, including annotated corpora and embedding models, under the framework of Petro NLP [Cordeiro et al. 2024a]. This process was guided by the PetroKGraph Ontology, a geological ontology populated with classes, instances, subclasses, and relations extracted from technical documents and compiled by geoscientists. We used the PetroKGraph Ontology to generate the text dataset, and it used RDF to store the information.

4. Methodology

In order to extract contexts from PetroKGraph, a well-structured data extraction pipeline is required.[2] Figure 1 provides a systematic way to navigating, querying, and transforming data embedded in the RDF framework. The procedure begins with understanding the RDF graph structure, which is required to ensure that subsequent extraction methods are successful. By establishing the foundation with a thorough understanding of RDF principles, schema analysis, and graph exploration, we can efficiently construct a pipeline.

We start by understanding the RDF graph structure of PetroKGraph using Protégé software[3] to visualize and explore its nodes, relationships, and entities. Also, we employ SPARQL to query and analyze the graph's data patterns. We find that the PetroKGraph has 2,069 entities and 26 relations. Some of the entities and relations include: $basins$, $fields$, $well$, $texture$, $lithostratigraphic\_unit$, $constituted\_by$, $located\_in$, $crosses$, has_age, $participates\_in$ and $part\_of$.

To extract and analyze data from an RDF graph, we first set up a suitable environment by installing Python and essential libraries, specifically *rdflib*. After installation,

[2]We use *contexts* as textual statements encoding the knowledge from a KG in natural language.

[3]https://protege.stanford.edu/

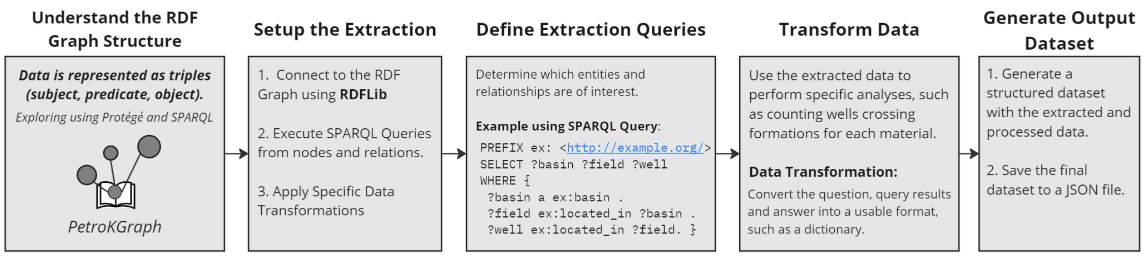
Understand the RDF Graph Structure
Data is represented as triples (subject, predicate, object).
Exploring using Protégé and SPARQL
PetroKGraph
Setup the Extraction
1. Connect to the RDF Graph using RDFLib
2. Execute SPARQL Queries from nodes and relations.
3. Apply Specific Data Transformations
Define Extraction Queries
Determine which entities and relationships are of interest.
Example using SPARQL Query:
PREFIX ex: <http://example.org/>
SELECT ?basin ?field ?well
WHERE {
?basin a ex:basin .
?field ex:located_in ?basin .
?well ex:located_in ?field. }
Transform Data
Use the extracted data to perform specific analyses, such as counting wells crossing formations for each material.
Data Transformation:
Convert the question, query results and answer into a usable format, such as a dictionary.
Generate Output Dataset
1. Generate a structured dataset with the extracted and processed data.
2. Save the final dataset to a JSON file.

Figure 1. Pipeline for Data Extraction from RDF Knowledge Graph - PetroKGraph

we use *rdflib* to load the RDF graph into the environment. With a clear grasp of the RDF triples, key classes, properties, and relationships, we can now proceed to design and implement the pipeline. In this implementation phase, we determine which entities and relationships are of interest to effectively query, extract, and transform data. We write SPARQL queries and algorithms to extract relevant data from the PetroKGraph (Fig. 1).

We start transforming data by running SPARQL queries and algorithms as shown in next section 5. A dataset of context, question, and answer is created by these queries, which retrieve pertinent information based on criteria for Portuguese.

5. Experimental Results

Algorithm 1 is designed to process lithostratigraphic units or geological formation data, focusing on mapping materials to the formations they constitute. This is particularly useful in geological contexts, where the algorithm iterates through formations, extracts relevant material information, and automatically generates questions based on the relationships between geological formations and their materials.

Algorithm 1 Query to retrieve the materials that constitute each lithostratigraphic unit

```
1:  materials_to_formations ← {}
2:  for all formacoes, info in formations.items() do
3:      if info[“constituted_by”] is not empty then
4:          formation_name ← info[“labels”][0] if info[“labels”] else formacoes
5:          for all material in info[“constituted_by”] do
6:              material_id ← material.split(“#”)[1]
7:              material_name ← str(g.value(URIRef(namespace + material_id), rdfs.label))
8:              if material_name not in materials_to_formations then
9:                  materials_to_formations[material_name] ← []
10:             end if
11:             materials_to_formations[material_name].append(formation_name)
12:         end for
13:     end if
14: end for
```

After processing all formations, the code iterates over the $materials\_to\_formations$ dictionary as shown in Algorithm 2. For each material, it constructs a question about which lithostratigraphic units (formations) are constituted by that material. The question, along with the list of formations (as the answer) and a contextual statement, is stored in a questions list and saved in the dataset’s JSON file.

Algorithm 2 Generate questions for lithostratigraphic units by material

```
1: for material_name, formation_name in materials_to_formations.items() do
2:     all_formations_names ← ", ".join(formation_name)
3:     formations_list ← [str(formacao) for formacao in formation_name]
4:     questions.append({
5:         "question": f"Que unidades litoestratigráficas são constituídas por {material_name}?",
6:         "answer": formations_list,
7:         "context": f"As unidades litoestratigráficas constituídas pelo {material_name} são:
           {all_formations_names}."
8:     })
9: end for
```

Additionally, we used a different type of query, such as Algorithm 3, which retrieves information about basins from a SPARQL endpoint, basins ($?basin$) and lithostratigraphic units ($?lithostratigraphic\_unit$) intersected by wells located in those basins. After processing all formations and basins, populates a list of question-answer pairs about the basins containing a specific geological formation.

Algorithm 3 SPARQL query to retrieve basins with specific lithostratigraphic units

```
1: SELECT ?basin ?litho_unit
2: WHERE {  ?basin rdf:type ont:basin .
3:          ?litho_unit rdf:type ont:lithostratigraphic_unit .
4:          ?well ont:crosses ?litho_unit .
5:          ?well ont:located_in ?basin .  }
```

Table 1 presents a few text templates designed to extract specific information from PetroKGraph using the data extraction pipeline in Fig. 1. Each row represents a specific query formulated to obtain details about different aspects of the PetroKGraph, such as field locations and geological age of formations. This table illustrates the effectiveness of the queries and the applicability of the extraction method for collecting pertinent data.

Table 1. Text templates developed to extract information from PetroKGraph

id	Query	Relation
1	Onde está localizado o campo {campo_name}?	located_in
2	Onde atravessa o poço {well_name}?	crosses
3	Qual é a idade geológica de {formation_name}?	has_age
4	Qual é a entidade que faz parte de {formation_name}?	part_of

6. Conclusions

We develop a dedicated pipeline for extracting and analyzing complex geological data by utilizing the pre-existing PetroKGraph. We generated a dataset of 10,219 items, each consisting of a question, an answer, and a context. It will be used in a question answering (Q&A) system based on Retrieval-Augmented Generation (RAG), both for ingestion and for evaluating its performance in a closed-domain setting. This system will combine efficient data retrieval with contextual answer generation, enhancing the ability to provide accurate and relevant responses based on the information extracted from the PetroKGraph. Integrating this dataset into the Q&A system will enable smoother and more precise text generation from LLMs.

Acknowledgments

This work was supported in part by the Brazilian National Agency of Petroleum, Natural Gas and Biofuels (ANP) and the Brazilian National Council for Scientific and Technological Development (CNPq) under Grant 140506/2023-3.

References

[Ali et al. 2022] Ali, W., Saleem, M., Yao, B., Hogan, A., and Ngomo, A.-C. N. (2022). A survey of rdf stores & sparql engines for querying knowledge graphs. *The VLDB Journal*, pages 1–26.

[Colas et al. 2022] Colas, A., Alvandipour, M., and Wang, D. Z. (2022). Gap: A graph-aware language model framework for knowledge graph-to-text generation. *arXiv preprint arXiv:2204.06674*.

[Cordeiro et al. 2024a] Cordeiro, F. C., da Silva, P. F., Tessarollo, A., Freitas, C., de Souza, E., Gomes, D. d. S. M., Souza, R. R., and Coelho, F. C. (2024a). Petro nlp: Resources for natural language processing and information extraction for the oil and gas industry. *Computers & Geosciences*, page 105714.

[Cordeiro et al. 2024b] Cordeiro, F. C., Silva, P. F. d., Gomes, D. d. S. M., Souza, R. R., Coelho, F. C., and Ell, B. (2024b). Petro kgraph: A methodology for extracting knowledge graph from technical documents-an application in the oil and gas industry. *Available at SSRN 4776804*.

[Elbassuoni et al. 2010] Elbassuoni, S., Ramanath, M., Schenkel, R., et al. (2010). Searching rdf graphs with sparql and keywords. *IEEE Data Eng. Bull.*, 33(1):16–24.

[Harris et al. 2013] Harris, S., Seaborne, A., Prud'hommeaux, E., et al. (2013). Sparql 1.1 overview. *W3C recommendation*.

[Koncel-Kedziorski et al. 2019] Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M., and Hajishirzi, H. (2019). Text generation from knowledge graphs with graph transformers. *arXiv preprint arXiv:1904.02342*.

[Lewis 2019] Lewis, M. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

[Manola et al. 2004] Manola, F., Miller, E., McBride, B., et al. (2004). Rdf primer. *W3C recommendation*, 10(1-107):6.

[Mizell et al. 2014] Mizell, D., Maschhoff, K. J., and Reinhardt, S. P. (2014). Extending sparql with graph functions. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 46–53. IEEE.

[Peng et al. 2017] Peng, P., Zou, L., and Qin, Z. (2017). Answering top-k query combined keywords and structural queries on rdf graphs. *Information Systems*, 67:19–35.

[Raffel et al. 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

[Ribeiro et al. 2020] Ribeiro, L. F., Schmitt, M., Schütze, H., and Gurevych, I. (2020). Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv:2007.08426*.

[Yuan and Färber 2023] Yuan, S. and Färber, M. (2023). Evaluating generative models for graph-to-text generation. *arXiv preprint arXiv:2307.14712*.