Translating data science queries from natural language into graph analytics queries using NLDS-QL

Genoveva Vargas-Solar, Karim Dao, Javier Espinosa-Oviedo

Translating data science queries from natural language into graph analytics queries using NLDS-QL

Genoveva Vargas-Solar[1,*], Karim Dao[2] and Javier A. Espinosa-Oviedo[1,3]

[1]CNRS, Univ Lyon, UCBL, INSA Lyon, LIRIS, UMR5205, Villeurbanne, France
[2]University Paris Dauphine, Tunis, Tunisia
[3]CPE Lyon, Villeurbanne, France

Abstract

This paper introduces NLDS-QL[1], a translator of data science questions expressed in natural language (NL) into data science queries on graph databases. Our translator is based on a simplified NL described by a grammar that specifies sentences combining keywords to refer to operations on graphs with the vocabulary of the graph schema. This paper shows NLDS-QL in action within a scenario to explore and analyse a graph base with patient diagnoses generated with the open-source Synthea.

Keywords

data science queries, graph analytics, natural language processing, graph stores

1. Introduction

The volume of connected data, often modelled as graphs, has grown exponentially. The availability of these graphs data collections has been democratised through social networks and knowledge graphs used to explore content (e.g. scientific papers, clinical cases). Although this accessibility is promising, it introduces a barrier for non-experts, who have to familiarise with the nature of the data, the way they have been represented in the database and the specific query languages or user interfaces to access them.

Besides, the emergence of data science has brought a new type of 'complex' queries embodying a data analysis scenario. A data science query generally refers to a workflow of tasks including exploration, data cleaning and preparation, sampling and analysis. These workflows include visualisation and evaluation tasks that involve the calculation of scores and metrics. Implementing these workflows is a challenge even for engineers and data scientists. In most cases, users should have advanced skills in querying and analysing the data according to their needs and the type of search questions to be answered. Requiring formal and technical expertise from non-computer scientists is not obvious or reasonable.

In our work, the goal is to identify important research questions, formulated by non-technical experts over data collections. Allowing their expression in natural language (NL) would optimize data accessibility. However, this facility implies complex NL analysis, particularly when such questions are intended to be transformed into sophisticated workflows. Thus, to achieve our goal, we address the problem under a reverse-engineering strategy: we build a simplified natural language (NL) grammar and map NL data science questions to graph data science queries as those proposed by Neo4J DS templates. The users can then express their questions using this simplified NL with sentences that correspond to the DS Neo4J templates, seeing and *assessing the results and proposing new questions over an enriched vocabulary that can extend the NL query language treated by our tool.*

Contribution We propose NLDS-QL, a semi-automatic and evolutive interface for processing experts' NL questions on a given vocabulary. It derives data science query templates that program the answers to these questions, and offers a conversational evaluation of results and adapted vocabulary. NLDS-QL is based on a simplified English NL that associates a vocabulary based on graph schema keywords to refer to operations on graphs (e.g., attributes describing the nodes, links, and associated labels). Depending on the type of queries to explore graphs or to analyse them, their expression in NL can yield to a more or less complex document.

NLDS-QL in action. NLDS-QL has been experimented in the context of medical diagnostics using the patient graph of the Synthea[1] data collection and using Neo4J for running translated queries. Assuming that they have

Published in the Workshop Proceedings of the EDBT/ICDT 2023 Joint Conference (March 28-March 31, 2023, Ioannina, Greece).
*Corresponding author.
genoveva.vargas-solar@cnrs.fr (G. Vargas-Solar); karim.dao@dauphine.tn (K. Dao); javier.espinosa@cpe.fr (J. A. Espinosa-Oviedo)
http://vargas-solar.com/ (G. Vargas-Solar); https://www.espinosa-oviedo.com/ (J. A. Espinosa-Oviedo)
0000-0001-9545-1821 (G. Vargas-Solar); 0000-0002-4015-195X (J. A. Espinosa-Oviedo)

[1]https://xilinx.github.io/graphanalytics/recom-tg3/synthea-overview.html

a collection of data corresponding to medical follow-ups, doctors (users) express the questions whose answers would be helpful for the elaboration of a diagnosis and their decision making. Questions described in written or spoken English can denote navigational, aggregation and data science queries requiring centrality and clustering algorithms to be expressed and answered. Given the ambiguity of the NL, the translation of questions can lead to several Cypher (data science) queries. So the use of NLDS-QL is proposed under a conversational pipeline where users acquainted with Cypher can choose the query that best corresponds to their expectations before executing them. Non-expert users can decide to let the system run different Cypher queries, analyse their results and then choose one or adjust their question.

The remainder of the paper is organised as follows. Section 2 synthesises the main families of works addressing NL to query languages translation and processing. Section 3 describes the general architecture of NLDS-QL. Section 4 describes the experimental scenario for NLDS-QL and the tasks proposed to users for testing it. Section 5 concludes the paper and discusses lessons learned from the implementation and experimention of our approach.

2. Related work

Existing work has addressed data mining using NL, but not yet the expression of data science questions. There are different approaches to developing a NL interface for database queries, in general for relational systems. Georgia Kutrika [1] describes the process of handling NL queries with a workflow that consists of three steps. Given a relational database and assuming knowledge of the schema vocabulary: 1. Analysis of the NL query expression; 2. Disambiguation and interpretation, which produces a set of ranked interpretations; 3. Finally, the translation into SQL and its execution. Three generations of NL Query to SQL Transformation systems can be identified [2], namely:

1. Keyword based i.e. information retrieval techniques to evaluate queries. For example, systems like Discover (query interpretations as subgraphs), DiscoverIR [3, 4], Spark (ranking and fast execution) [5].
2. NL processing based like NaLIR (parser) [6], ATHENA [7] (ontologies and mappings).
3. Machine translation using neural networks [8], like NL to SQL conversion as a language translation problem. The challenge is training a neural network on a large number of NL/SQL query pairs. Approaches like SQLNet [9], Hydranet adopt this strategy.

On the other hand, we identify two families of works concerning NL Graph Querying.

1. Approaches that address NL translation on SPARQL for knowledge graph queries, such as [8], concerning machine learning techniques (Tree-LSTM and neural networks), and methods based on grammar and logical predicates [10, 11].
2. Approaches that translate questions into structured queries using NL processing methods such as: named entity recognition, binary relationship (pattern) extraction, key entity identification, and relationship mapping to graph components [12].

The literature agrees that little work has addressed the issue of answer validation, i.e. how can a user confirm that the results match the query's intent? With the emergence of data science, two questions arise: How to express data exploration, cleaning and preparation, sampling and analysis, visualisation and metrics calculation? How to model and process research questions formulated by non-technical experts? How to allow their NL expression and translation into data science queries?

Our work proposes an interactive reverse-engineering method that highlights essential aspects to be considered when dealing with NL medical queries. It is the basis for a user-friendly interface adapted for the medical personnel.

3. NLDS-QL

The general process implemented by NLDS-QL is shown in Figure 1. The first two phases of our translation approach are devoted to analysing the NL query, which is expressed as a text (see the NL processing and NL parsing phases in Figure 1). The text can be written or defined as a voice message and transcribed into text. Therefore, the NL processing phases implement the classical text processing of syntactic analysis to produce an expression tree that represents the query (this is done by a parser as shown in Figure 1). The tree is then processed to produce one or more corresponding Cypher queries in the query generation phase (see the query generation phase in Figure 1). Finally, the queries are evaluated on Neo4J (see the query evaluation phase in Figure 1).

Overview of NLDS-QL expressions. The expression of NLDS-QL questions is based on the way data science operations are applied on graphs in Neo4J. Neo4J defines a general template including several commands for expressing the execution of a DS query.

DS operations are generally applied on graph views created in memory from persistent graphs. The views require main memory space to be allocated for creating them and main memory resources for using algorithms with specific execution conditions expressed in parameters. Thus, Neo4J provides commands for performing these estimations and then calling DS operations with given parameters' values. Finally, DS operations can yield

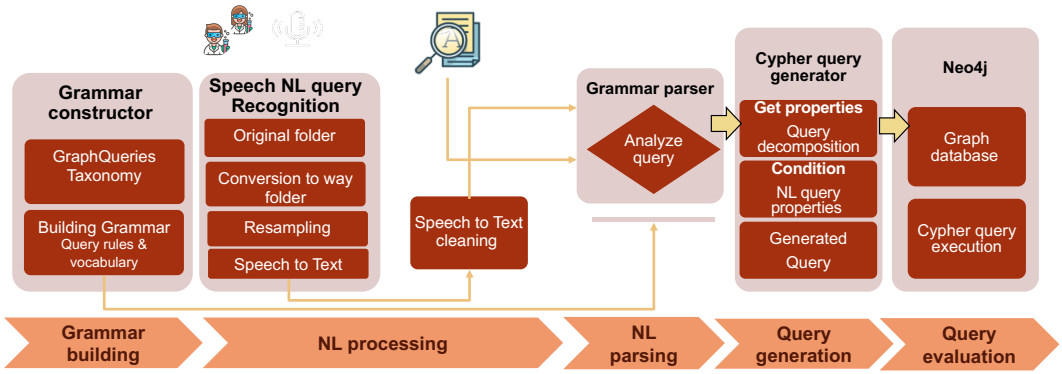

Figure 1: NLDS-QL general architecture

new graphs that can be named and persist or not. The creation of new graphs and whether to persist them is expressed as function call commands.

Consequently, the definition of data science questions in NL include expressions for specifying the commands specified in the Neo4J template. The most simple expression for defining a graph view and estimating the memory required assuming that it is stored in Neo4J and that the graph schema with the nodes and relations is available, is defined with the following English expression:

- *Create and estimate memory for the graph view* <subgraph-name> [*named as* < view name >] *with the node* < node name > *and the relationship* < relationship name > [*oriented*]

The data science operation task includes estimating the cost in memory of applying a graph data science algorithm on the graph, using the algorithm on the graph view. According to specific keywords, NLDS-QL can determine the type of algorithm that can be applied. Keywords like *most important, most popular, most influential* refer to centrality algorithms such as PageRank and Louvain and *classify, communities, group* can refer to clustering algorithms like Label Propagation, as illustrated by the following three questions:

- Q_1 : *Estimate the required memory for applying* < DS algorithm name> *on the graph view* < view name >
- Q_2 : *Find the most important/most popular* < node name > *with* < relation name > [*in the graph* < graph name >] *with* < number of iterations > *maximum of iterations and with a damping factor* < floating number >
- Q_3 : *Classify/Find groups/communities of* < node name > *within the view* < view name > *with relation* < relation name > *with* < number of iterations > *maximum of iterations*

4. Applying NLDS-QL for exploring a medical graph

We set up an experiment to validate our approach. Therefore we use the patient part of the Synthea Generic study. The Synthea's Generic Graph[2] models various diseases conditions that contribute to the medical history of synthetic patients 800K vertices and nearly 2000K edges.

Querying graphs is based on navigational queries, which retrieve information already "contained" in a graph. In the Synthea graph, it is possible to ask simple queries like : How many patients are there in the Synthea study? Which allergies are identified in the Synthea study patients? But it is also possible to go further and ask analytical type questions that involve classification tasks such as What are the most frequently prescribed drugs for patients in the Synthea study? Answering this type of query involves performing a sequence of tasks ordered in a pipeline, which we call a data science query.

Use case The use case is based on the Synthea **patients graph** shown in Figure 2. It describes the immunisations, allergies, conditions, studies, procedures and care plans of patients. Each entity and its relations are characterised by properties that describe them. The patient graph has approximately 100 thousand nodes and 37 thousand relations stored on Neo4J.

The use case of NLDS-QL on the Synthea patients graph is based on a conversational pipeline where expert and non-expert users can ask questions to start exploring the graph (see Figure 3). The use case environment initially shows the Synthea patients graph, and users can ask for details about the description of the graph, like the number of nodes and relations. Then, the user can ask a question. The system generates one or several queries,

[2]https://xilinx.github.io/graphanalytics/recom-tg3/synthea-overview.html

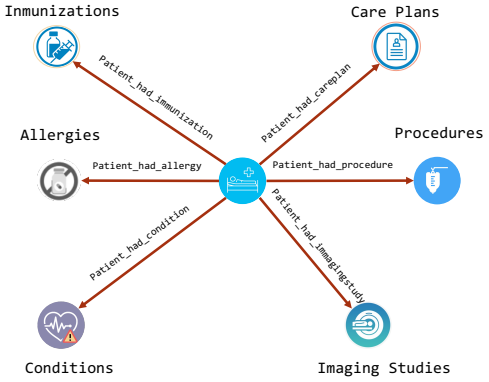

Figure 2: Synthea patients' graph

and then the user can either choose one or several queries to be adjusted or executed and then modified (see right side of Figure 3). For every choice, the user can evaluate the system's performance with stars that show the degree of satisfaction.

For exploring the Synthea graph, the use case proposes a set of queries that include navigational queries of the type selection, projection, aggregation.

Selection. *Find the Medications for which the DESCRIPTION is Lisinopril 10 MG Oral Tablet and the REASON of the DESCRIPTION is Hypertension.*

```
MATCH (n:Allergies)
return n.DESCRIPTION
```

Projection. *Which is the birthplace of the PATIENTS in the study?*

```
MATCH (n:Patients)
return n.BIRTHPLACE
```

Selection and Projection. *Find the Encounters DESCRIPTION node where the DESCRIPTION of the drugs is Amlodipine 5 MG Oral Tablet.*

```
MATCH
  (n:Encounters)-[*]->
  (m:Medications {
    DESCRIPTION:
      'Amlodipine 5 MG Oral Tablet'
  })
return n.DESCRIPTION, m.DESCRIPTION
```

Aggregation. *How many patients are caucasian?*

```
MATCH (n:Patients {RACE:'white'})
return count(n)
```

For data science queries, the use case shows NLDS-QL questions that refer to centrality type operations. Note that the translation is quite complex as it involves:

- Specifying a graph view from the patient graph, as Neo4J works with graph views stored in RAM when data science algorithms are applied.
- Then it is possible to generate two queries that call the page rank algorithm to process the keyword "most important" with the possibility to make the view persistent and consider the constraints related to the parameters of the Pagerank algorithm.

Centrality.
Find the most popular Encounters for Medications in the graph.

```
MATCH
  (n:Encounters)-
  [r:ENCOUNTER_FOR_MEDICATION]-()
  with n,count(*) as degree
  return id(n), degree
ORDER BY (degree) DESC
```

Find the most important Drugs prescribed for the PATIENT with a maximum of 25 iterations and a damping factor of 0.60.

```
CALL gds.graph.create(
    'my_graph',
    'Medications', {
        PATIENT_HAS_MEDICATION: {
            orientation: 'NATURAL'
        }
    })

CALL gds.pageRank.write.estimate(
    'my_graph', {
        writeProperty: 'pageRank',
        maxIterations: 25,
        dampingFactor:0.60
    })
YIELD nodeCount, relationshipCount,
      bytesMin, bytesMax,
      requiredMemory

CALL   gds.pageRank.
       stream('my_graph')
YIELD  nodeId, score
RETURN gds.util.
       asNode(nodeId).name AS name,
       score
ORDER BY score DESC LIMIT 10
```

In this example, NSDL-QL generates the template that includes first the graph "my_graph". Then it computes the estimation of required memory, number of nodes and relations, minimum and maximum bytes that will yield the resulting graph when executing PageRank with the

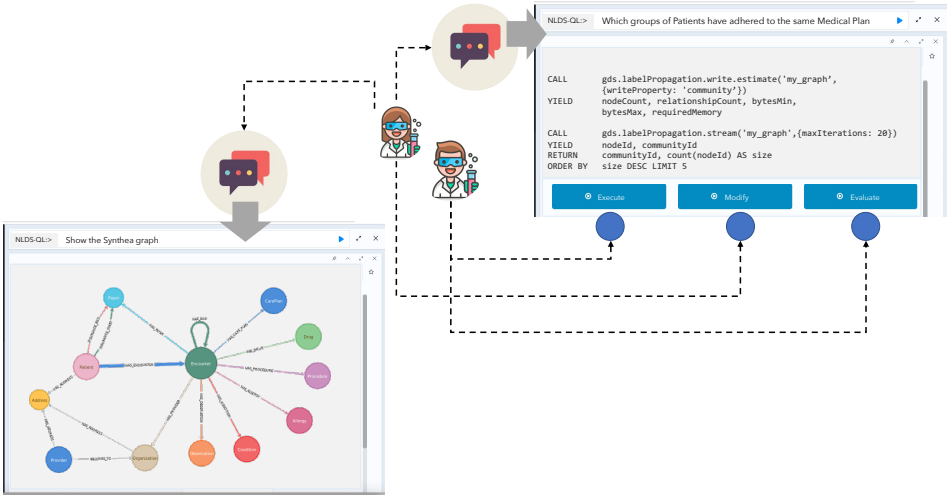

Figure 3: Use case conversation pipeline

specified parameters. Then the call to the algorithm with the result format with the top 10 nodes associating each node with its score.

Community detection. The translation also involves several operations as described in the definition of data science queries.

Get the subgroup of Patients who have PATIENT_HAS_CAREPLAN in the graph with max iterations 20

```
CALL gds.labelPropagation.
     write.estimate(
     'my_graph', {
        writeProperty: 'community'
     })

YIELD nodeCount, relationshipCount,
      bytesMin, bytesMax,
      requiredMemory

CALL gds.labelPropagation.
     stream('my_graph',
     {maxIterations: 20})
YIELD     nodeId, communityId
RETURN    communityId,
          count(nodeId) AS size
ORDER BY  size DESC LIMIT 5
```

5. Conclusion and Results

This paper introduced NLDS-QL and showed through a use case how to map NL data science questions (using an adapted vocabulary) to Neo4J data science query templates. The use case aimed at querying and analysing a graph in the medical domain. Users with medical and non-medical backgrounds can define a sequence of natural language queries executed step by step to explore the graph, as in data science questions. Thereby users can acquire an understanding of medical prescriptions proposed to patients by classifying their treatment, their physiological characteristics to better understand how diseases are diagnosed and treated according to patients conditions. In this way, we showed the essential aspects of a data science query template expressed in NL.

The approach is flexible and can be enhanced for processing documents with richer NL vocabulary and more complex templates. The intervention of a human in handling natural language queries calls for the design of an interactive strategy based on conversation. We have started to design a more evolved conversational interface considering human in the loop and user profiling techniques.

References

[1] G. Koutrika, The rise of intelligent data assistants: Democratizing data access - keynote, 4th International Workshop on Big Data Visual Exploration and Analytics - EDBT/ICDT Workshops (2021).

[2] O. Gkini, T. Belmpas, G. Koutrika, Y. Ioannidis, An in-depth benchmarking of text-to-sql systems, in: Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 632–644.

[3] D. Song, F. Schilder, C. Smiley, C. Brew, T. Zielund, H. Bretz, R. Martin, C. Dale, J. Duprey, T. Miller,

et al., Tr discover: A natural language interface for querying and analyzing interlinked datasets, in: International Semantic Web Conference, Springer, 2015, pp. 21–37.

[4] S. Lakhanpal, A. Gupta, R. Agrawal, Discover trending domains using fusion of supervised machine learning with natural language processing, in: 2015 18th International Conference on Information Fusion (Fusion), IEEE, 2015, pp. 893–900.

[5] A. Thomas, Natural Language Processing with Spark NLP: Learning to Understand Text at Scale, " O'Reilly Media, Inc.", 2020.

[6] F. Li, H. V. Jagadish, Nalir: an interactive natural language interface for querying relational databases, in: Proceedings of the 2014 ACM SIGMOD international conference on Management of data, 2014, pp. 709–712.

[7] D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, F. Özcan, Athena: an ontology-driven system for natural language querying over relational data stores, Proceedings of the VLDB Endowment 9 (2016) 1209–1220.

[8] A. Baevski, H. Zhou, A. Mohamed, M. Auli, wav2vec 2.0: A framework for self-supervised learning of speech representations, arXiv preprint arXiv:2006.11477 (2020).

[9] X. Xu, C. Liu, D. Song, Sqlnet: Generating structured queries from natural language without reinforcement learning, arXiv preprint arXiv:1711.04436 (2017).

[10] S. Liang, K. Stockinger, T. M. de Farias, M. Anisimova, M. Gil, Querying knowledge graphs in natural language, Journal of big Data 8 (2021) 1–23.

[11] Z. Liu, X. Xia, A. E. Hassan, D. Lo, Z. Xing, X. Wang, Neural-machine-translation-based commit message generation: how far are we?, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 2018, pp. 373–384.

[12] E. Oro, M. Ruffolo, A natural language interface for querying rdf and graph databases, Consiglio Nazionale delle Ricerche Istituto di Calcoloe Reti and Alte Prestazioni (2015).