Scalable No-Code Knowledge Graph Exploration and Querying with SemSpect

Thorsten Liebig[1,†], Michael Opitz[1,†], Vincent Vialard[1,*,†] and Maximilian Wenzel[1,†]

[1]derivo GmbH, Olgastr. 143, 89073 Ulm, Germany

Abstract

How can we gain meaningful insight into large RDF or labeled property graphs? Which groups of nodes are connected, what relationships does a single node have? This demo is addressing these challenges and will demonstrate SemSpect, a new tool which uses visual aggregation to solve the hairball problem of graph visualization. The new approach enables users to explore large knowledge graphs from the meta level down to all node details without any prior idea of the data. We will demonstrate how to build sophisticated queries in a data-driven manner without any query language skills.

Keywords

Knowledge Graphs, RDF, LPG, Graph Visualization, Graph Exploration

1. Motivation

Knowledge Graphs are a powerful way to represent and query domain-specific information, but they can also be challenging to explore and understand, especially when they are large and complex. How to gain insight into such data graphs compiled from many different sources? How to discover the structure and identify data flaws even without any prior idea of the data or any query language skills? Well-known graph visualizations and interactive renderings are often mentioned as a solution to these tasks. Every Knowledge Graph user had experiences with visualizations that render data graphs in terms of single nodes and edges, allowing to interactively add related data nodes step by step to one or more already visible nodes. This is data-driven and great for small graphs, but already fails for medium sized graphs [1]. We can identify three main issues for this particular approach if the graphs become increasingly large: First, it quickly leads to visual clutter and information overload as the graph grows. Second, while force-feedback layouting alleviates some of the problems, it can cause disorientation due to the necessary rearrangement of nodes. Lastly, it can miss important insights, as it only shows a local and partial view of the graph structure.

As an example, let us analyze and query the Panama Papers, which represents a network that reveals offshore business links of wealthy and prominent individuals that may have been kept

SEMANTICS 2023 EU: 19th International Conference on Semantic Systems, September 20-22, 2023, Leipzig, Germany

*Corresponding author.

†These authors are listed in alphabetical order and have contributed equally.

liebig@derivo.de (T. Liebig); opitz@derivo.de (M. Opitz); vialard@derivo.de (V. Vialard); wenzel@derivo.de (M. Wenzel)

0000-0002-2810-7315 (T. Liebig)

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3526/paper-07.pdf

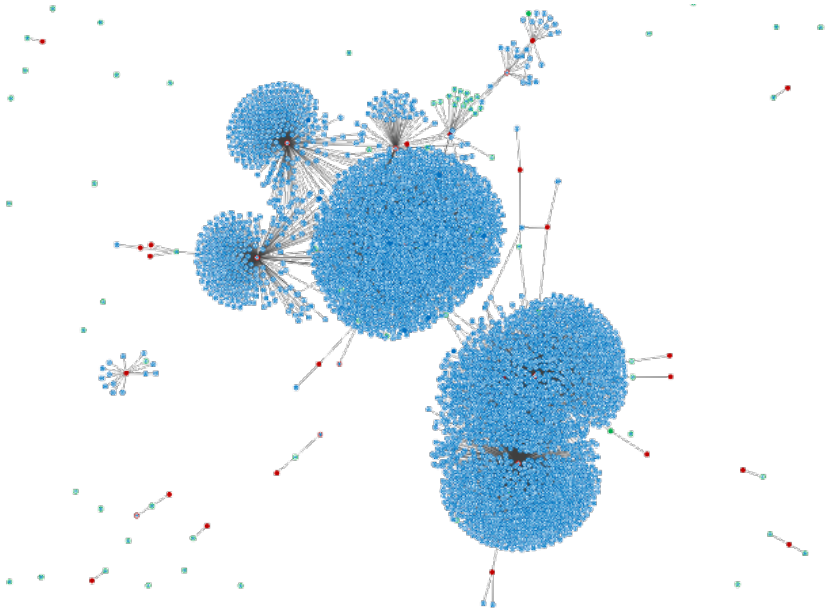

Figure 1: Exemplary study of the Panama Papers dataset showing entities in Germany, their addresses, and the officers that are registered at those addresses using traditional network rendering.

private, e.g., for reasons of tax evasion. In this case we draw on the work of many journalists of the International Consortium of Investigative Journalists (ICIJ[1]) who have built a Knowledge Graph by extracting selected data out of offshore leaks such as the Panama, Paradise and Pandora paper leaks. They searched through thousands of e-mails, contracts and bank account statements to build a graph consisting of 2 million nodes and over 3 million relationships[2].

Let us examine the officers that share an address with an entity in Germany. For this, we have to filter for the German-based entities first, find their addresses, and in turn find the officers of these addresses. Figure 1 depicts an excerpt of the resulting network rendering showing entities from Germany (green dots) their addresses (red dots) and the officers at these addresses (blue dots). As can be seen, there is a high number of officers that share one and the same address. However, without zooming in, it is difficult to determine how many officers there are exactly and to which addresses they are specifically linked to. Visualizing graphs as node-link networks helps to reveal the graph topology as it shows clusters of nodes, isolated sub-networks, and disconnected nodes. But when the exploration query involves more hops and nodes, the data becomes hard to comprehend due to visual clutter.

2. Querying and Visual Exploration Beyond the Hairball

To address this problem, we present SemSpect[3], a graph exploration tool that aims to achieve a better scalability while preserving an intuitive UI and offering a good expressivity-usability

[1]https://www.icij.org/
[2]download at https://offshoreleaks.icij.org/
[3]https://www.semspect.de/

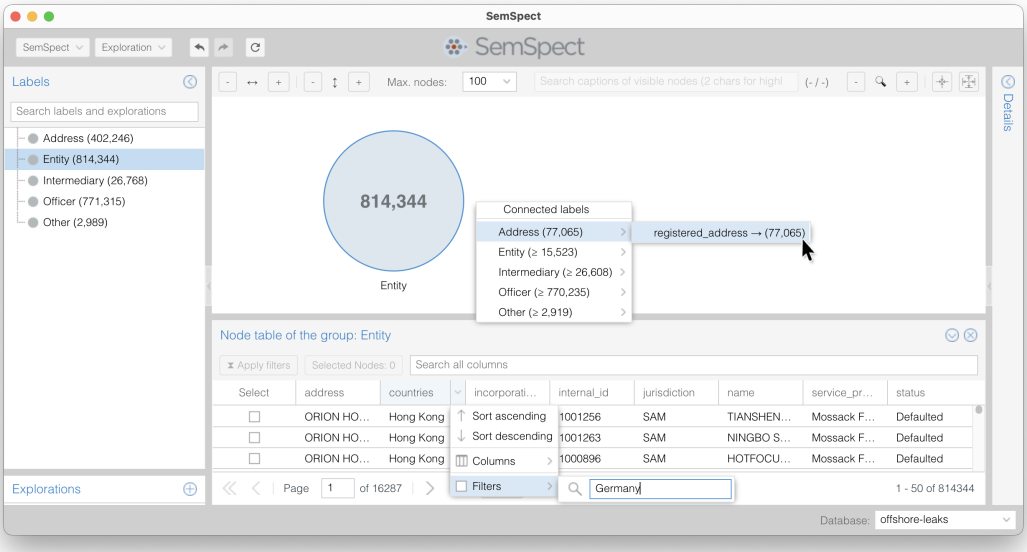

Figure 2: Exploration of the Panama Papers Knowledge Graph with SemSpect. Top right: exploration canvas with ”Entity” group and exploration menu. Bottom right: tabular view of ”Entity” nodes.

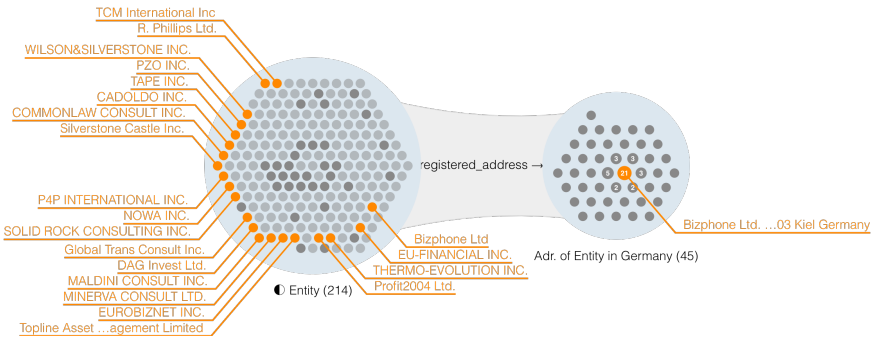

Figure 3: First exploration and filtering step of study of Figure 1 showing the entities in Germany (left group) and their addresses (right group).

balance. It supports both RDF graph and Labeled Property Graph (LPG) models, grouping nodes by their labels or class types. SemSpect enables users to explore these groups selectively following the ”overview first and details on demand” principle. It also allows users to construct and export expressive queries without writing any SPARQL or Cypher.

For our example of entities from Germany, we start to build the query by dragging ”Entity” into the exploration canvas as depicted in Figure 2. A double click on the resulting group displays the list of connected node types as well as the connecting relationships for selection. Optionally, we can display the list of nodes of a selected group via a tabular view at the bottom and filter particular attribute values. For the entities from Germany we filter the ”countries” attribute to ”Germany” as depicted in Figure 2. Remark: Nodes could also be selected manually for a more focused exploration.

Applying this filter leaves 214 nodes which are related to 45 addresses as shown in Figure 3. The numbers in the individual address nodes indicate the number of related entities in the left group. An interesting finding is that 21 entities in Germany share one and the same address in Kiel. SemSpect can highlight these connections with tags as can be seen in Figure 3, thereby taking advantage of the compactness of its exploration layout over the more informative tabular view. Finally, the answer to our main question is provided in Figure 4. There are 4647 officers in the Panama Paper dataset that are registered at addresses of entities in Germany. The right group in Figure 4 shows the three addresses that have the most registered officers, which are highlighted with tags.

2.1. Query Expressivity

By reading an exploration backwards, we get an informal idea of the query expressivity in SemSpect: a group of nodes is basically defined by a restricted path (or tree of paths). Groups of nodes can be restricted by their types (conjunction / disjunction / difference), their attributes (conjunction of value restrictions), or even by a manual selection of nodes. They may be connected to other recursively defined groups over a chosen relationship type that can be restricted by its attributes (conjunction of value restrictions) and its cardinality (number of predecessors). The possibility to define new types based on an exploration group expands the expressivity considerably by implicitly adding a form of recursion to the type filtering.

3. Technical Description

SemSpect is a client-server application with an HTML5/JavaScript UI and a Java REST backend. As storage component, we provide a Neo4j backend for Labeled Property Graphs (LPG) and, on the other hand, an in-memory RDF backend which supports RDFS and `owl:inverseOf` entailment. Moreover, the RDF version allows users to explore annotations on knowledge graph edges, which have been reified according to RDF/OWL or specified in the RDF-star format.

3.1. Neo4j Backend

The SemSpect Neo4j backend is a plugin for Neo4j server as well as a Graph App for Neo4j Desktop. This close integration allows efficient access to Neo4j data structures, speeding up the

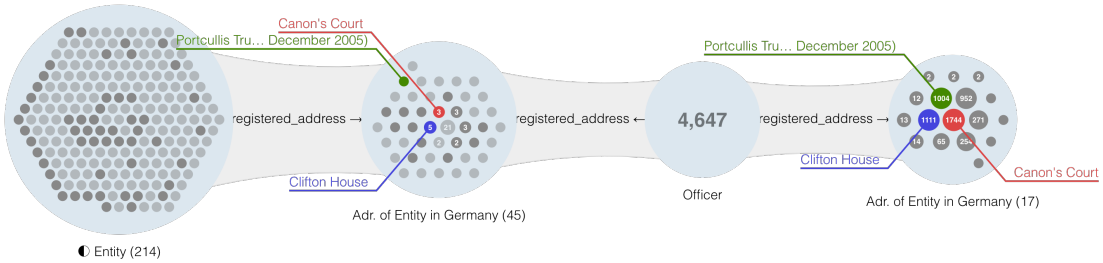

Figure 4: Study of Figure 1 with tags for the three topmost referenced addresses by officers.

initialization process that scans the database to gather information about the structure of the graph. This information is used to reconstruct the implicit graph model, infer the label hierarchy and map user request to Cypher queries. The frontend communicates with the backend via bolt in a "REST over Bolt" [2] fashion to share as much code as possible with other implementations.

3.2. In-memory RDF Backend

Our RDF version of SemSpect uses an in-memory Java backend that is based on index structures specifically designed to efficiently process frontend requests. These include a variation of Bitmap Triple (BT) indices [3] that extend the compact triple data structure of the RDF compression format HDT [4]. These low-memory data structures were fine-tuned to achieve efficient access to the graph data. We start with a set of RDF dump files that include the corresponding schema/ontology and compute a compact representation of the data (remark: in contrast to our Neo4j based approach, the subtype definitions currently have to be explicitly specified in the schema/ontology). Subsequently, we build the query indexes which are eventually written to disk. The indices encode a snapshot of the given data, i.e., if the data changes, all associated structures have to be generated anew. However, once all indices have been generated, they can be efficiently loaded into main memory for an exploration with SemSpect.

4. Demo

At SEMANTICS 2023 we will introduce the new, high-performance in-memory RDF backend for SemSpect to the public. In our demo, we will showcase SemSpect with different datasets in terms of their domain and size. We will guide attendees to use SemSpect themselves and conduct their own explorations and queries. Moreover, we will be ready to load ad-hoc data provided as either a Neo4j or RDF dump to demonstrate and experience the performance of both backend variants.

SemSpect is a commercial software, but we offer a feature-restricted free version and grant academic licenses on request.

References

[1] V. Yoghourdjian, D. Archambault, S. Diehl, T. Dwyer, K. Klein, H. C. Purchase, H.-Y. Wu, Exploring the limits of complexity: A survey of empirical studies on graph visualisation, Visual Informatics 2 (2018) 264–282.

[2] M. Opitz, C. Ranz, Bolt On Your Web App to Neo4j, Neo4j NODES Developer Conference, 2019. URL: https://www.youtube.com/watch?v=Tsa_d-V7Bb4.

[3] M. Wenzel, T. Liebig, B. Glimm, HDT Bitmap Triple Indices for Efficient RDF Data Exploration, in: The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18, Springer, 2021, pp. 109–125.

[4] J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, M. Arias, Binary RDF representation for publication and exchange (HDT), Journal of Web Semantics 19 (2013) 22–41.