Towards a Knowledge Graph Lifecycle: A pipeline for the population of a commercial Knowledge Graph

Umutcan Şimşek[1], Jürgen Umbrich[2], and Dieter Fensel[1]

[1] Semantic Technology Institute Innsbruck, University of Innsbruck
Technikerstrasse 21a,
6020 Innsbruck, Austria
{firstname.surname}@sti2.at
http://sti2.at
[2] Onlim GmbH
Telfs, Austria
juergen.umbrich@onlim.com

Abstract. This paper is a use case report for the population architecture of a commercial Knowledge Graph. We introduce our pilots and their focus within the MindLab Project, which aims to build Knowledge Graphs with a lifecycle-based approach to enable conversational agents. We describe and evaluate our pipeline for the first step of the lifecycle, namely Knowledge Creation. Our approach satisfies all defined requirements in terms of provenance tracking, scalability, and usability.

Keywords: Knowledge Graph · Data pipeline · Knowledge Creation

1 Introduction

Knowledge Graphs are an important means to provide large-scale integrated data to intelligent applications like conversational agents. Making a Knowledge Graph a useful resource requires to complete a set of tasks that comprise the *Knowledge Graph lifecycle* [4]:

Knowledge Creation	Semantic lifting and integration of heterogeneous data sources.
Knowledge Hosting	Storage of the knowledge in a suitable way respecting its nature and applications (e.g.,, a graph database, triple store).
Knowledge Curation	A lifecycle-based approach makes sure that the correctness and completeness of the Knowledge Graph satisfy the needs.
Knowledge Deployment	Consumption of the Knowledge Graph by various applications.

For each of these tasks, there are various approaches, mainly developed and applied in an ad hoc way. Even for the - seemingly straightforward - Knowledge Creation task, there are bureaucratic and technical challenges due to the variety and heterogeneity of data sources. In this paper, we address this very task by providing a holistic approach and architecture to populate a commercial Knowledge Graph based on heterogeneous data sources. The Knowledge Graph is built in the scope of the MindLab project[3], which aims to build industrial Knowledge Graphs to enable conversational agents in domains like e-tourism and beyond. In the remainder of the paper, first, we describe the use cases and their requirements that our Knowledge Graph must fulfill (Section 2). We describe our technical approach that provides a workflow for creating knowledge from heterogeneous sources and populating a Knowledge Graph (Section 3). We give a brief overview of the related work in Section 5. Last but not least, we evaluate our technical approach in terms of satisfying the requirements (Section 4) and conclude with final remarks and future directions in Section 6.

2 Use Case Description

Our general use case is to import and integrate various heterogeneous data sources into one coherent data model using a schema that is highly aligned to the schema.org, a de facto standard for annotations on the web [6]. The data sources itself can be of various sizes and contain dynamic or rather static data. Our use cases must track the provenance of the sources, such as the importing time, origin, and mapping specification.

2.1 Pilots and Data Sources

In total, we selected three pilots from the tourism domain in our project. The three pilots are 1) Seefeld, 2) Serfaus-Fiss-Ladis, and 3) Mayrhofen.

All pilots have in common that users will interact with a bot in natural language (either via text or voice) and inquire about information related to the tourism domain. The data itself is modeled and stored in the form of a Knowledge Graph, which should be exploited to get i) a better language understanding, ii) more precise answers (result presentation), and iii) data-driven and guided dialogs.

Additionally, all pilots will operate over a set of data sources that contain information about various accommodation providers and local businesses, events, sports activities (e.g., hiking, trails, skiing), and sports areas (e.g., ski resorts) or webcams. The core tourism data is provided by five different data providers. Feratel provides accommodation and event data, General Solutions geospatial data, Intermaps ski resorts and slopes, Outdooractive outdoor activity data (e.g., hiking tours), and Verkehrsauskunft Österreich transportation data. The data comes in various formats (e.g., JSON and XML) and is either supplied in the form of data dumps or via RESTful API requests.

[3] https://mindlab.ai

Some pilots will further connect the tourism data to public and open data sources, which contain additional common knowledge about some entities of the tourism data. For instance, DBpedia[4] and Wikidata[5] provide additional information about cities used in the pilots or descriptions about certain sports activities. Also, publicly available geodata will be used for more advanced geolocation related conversations (e.g., OpenStreetMap[6]). A prominent open-source geodata provider is OpenStreetMap[7].

The data sources can provide static information (e.g., geolocations, names, categories) but also dynamic information (e.g.,, the current weather, snow level, open ski lifts, hotel rates, or transport information). Our solution needs to handle and represent such dynamic information, either in the form of updates in the Knowledge Graph or by specifying services to derive the current and most up-to-date information.

2.2 Requirements

Data importers The requirements for the data importers are that they have to scale with the input data, be usable and easy expandable, and that they track the provenance for generating statistics and easier debugging of the importer pipeline.

R-DI-prov	the importers should track vital provenance and performance metrics that are used to filter subgraphs, generate KPIs, performance analysis, and assist debugging[8].
R-DI-scale	data importers have to scale w.r.t. the number and size of input sources. As such, the developed solutions should be executable in parallel setup.
R-DI-usability	data importers should be designed in a way that they are easy to extend and adapt to new data sources. Ideally, the importing pipeline can be configured and used without specific technical knowledge (e.g., programming language agnostic)

3 Technical Approach

In this section, we describe our technical approach to tackling the knowledge creation task while addressing the requirements described in Section 2.2. The major features of the architecture are:

– Storing data as named graphs by utilizing quads

[4] https://wiki.dbpedia.org/
[5] https://www.wikidata.org/wiki/Wikidata:Main_Page
[6] https://www.openstreetmap.org/
[7] https://www.openstreetmap.org/
[8] Also aligned with the Data on the Web Best Practices https://www.w3.org/TR/dwbp/

- Attaching provenance information based on the modeled provenance metadata (Section 3.1)
- Scalable mapper for RDF generation from heterogeneous sources
- Flexible, declaratively configurable workflow for managing new data sources to populate the Knowledge Graph

The following explains how these features are conceptualized and implemented. We first explain our provenance tracking approach, including our metadata. Afterward, we describe our implementation, MindLab Importer, for the overall knowledge graph construction process.

3.1 Provenance Tracking

The MindLab Knowledge Graph must be consumable by applications from different perspectives such as data from certain providers or a given set of geospatial areas (R-DI-prov). The MindLab Knowledge Graph will be constructed by integrating data from heterogeneous sources. In order to satisfy the R-DI-prov requirement, we use the named graph [1] approach[9]. For each quad in the Knowledge Graph, the fourth element is the context URI. The provenance information is attached to this context URI. Based on the requirements mentioned above, we identify named graphs per organization per IT solution provider. An example URL for DMO Mayrhofen organization and feratel IT solution provider is https://graph.mindlab.ai/tvb-mayrhofen/feratel.

Provenance Metadata The metadata used for describing the provenance is mainly based on two vocabularies, namely PROV-O [11] and schema.org [6]. PROV-O is an ontology that provides types and properties to create metadata about the provenance of anything, including information about entities, their creation or modification process, and parties involved. Schema.org also provides types and properties for describing datasets, which we adopt for the named graphs in the MindLab Knowledge Graph. The provenance metadata relates to PROV-O and schema.org, as described in Table 1.

Listing 1.1 is an excerpt of the provenance information for a named graph. The named graph is connected to the organization (i.e., DMO Seefeld) and the IT solution provider (i.e., feratel) through schema:provider and prov:wasAttributedTo properties.

3.2 MindLab Importer

In this section, we introduce the architecture of the MindLab Importer and its core components. Figure 1 provides an overview of the architecture. The central component of the MindLab Importer is the Import Manager that coordinates

[9] A comparison of different reification approaches can be found in [7]. We adopt the named graph approach since it has the least space complexity and is natively supported by triple store implementations (e.g., GraphDB) and SPARQL

MindLab KG	Provenance Model	Description
Graph	prov:Entity, schema:Dataset	A named graph in MindLab KG is an entity in PROV-O and a dataset in schema.org
Import	prov:Activity	An import represents and individual import process that creates a named graph for a given organization and IT solution provider
Wrapper	prov:SoftwareAgent	A wrapper is a software agent that generates RDF from raw data.
Organization	prov:Organization, schema:Organization	The organization who provides the data.

Table 1. The mapping between MindLab Knowledge Graph concepts and PROV-O

```
<https://graph.mindlab.ai/tvb-seefeld/feratel> schema:creator
↪  mindlab-organization:mindlab ;
    schema:provider mindlab-organization:tvb-seefeld ;
    schema:publisher mindlab-organization:mindlab ;
    prov:wasAttributedTo mindlab-agent:feratel-wrapper ;
    prov:wasGeneratedBy mindlab-import:e87c7fb0-3c34-11e9-8de2-5520c2d355b7 .


mindlab-import:e87c7fb0-3c34-11e9-8de2-5520c2d355b7 a prov:Activity ;
    prov:endedAtTime
↪  "2019-03-01T15:15:54.794Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
    prov:startedAtTime
↪  "2019-03-01T15:05:00.202Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
    prov:wasAssociatedWith mindlab-agent:feratel-wrapper  .
```

Listing 1.1: An example provenance information attached TVB Seefeld and feratel named graph

the import process. The data sources from which the Knowledge Graph is populated are registered via a web interface to the Import Manager. When an import starts, the Import Manager uses the source-specific information (e.g., configuration required to access an external source) to retrieve the raw data and passes it to the mapping service. The service offers a scalable RML [3] mapper implementation [10] [13] to the Import Manager[11]. The RDF data generated based on schema.org is then returned to the Import Manager. The mapper service caches mapping files when needed, in order to improve the overall performance. The Import Manager pushes mapped data to its designated named graph in the Knowledge Graph. After the import is complete for a source, the Import Manager attaches the provenance information.

[10] https://github.com/semantifyit/RocketRML

[11] Depending on the IT solution provider, the mapping service may be replaced with other means of generating RDF

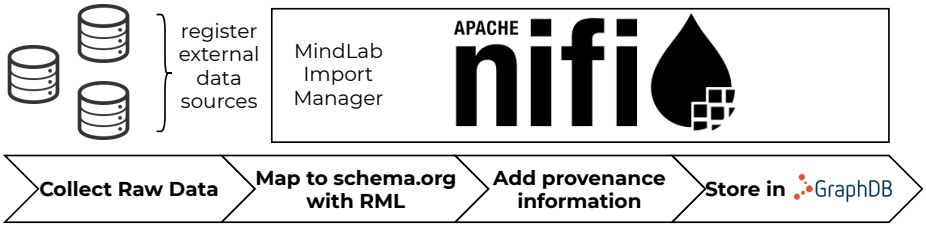

Fig. 1. MindLab Importer Architecture

The import manager is implemented with the Apache NiFi[12] data flow management tool. NiFi offers a UI for defining data flows in a drag-and-drop manner, alongside an API to create and manage such flows programmatically. It has many features like load balancing, buffering, guaranteed delivery and scheduling out of the box. It offers built-in processors for many tasks (e.g., connectors for different database solutions) and generic HTTP processors for accessing external services (e.g., our mapper service).

A NiFi workflow consists of connected units called processor. A processor typically receives a flow file, uses or manipulates it in some way and passes a new flow file to another processor. A subset of connected processors can be placed in containers called process groups. The Import Manager contains one process group for each distinct IT Solution provider and a single NiFi processor per organization per IT solution provider that acts as an initiator. This provides flexibility and scalability for the creation of new import tasks to some extent. Once a flow for an IT solution provider is created, a new organization using existing IT solution providers can create their own import task by just entering the necessary configuration (e.g., organizations API key) through the registry. The registry interface takes care of the generation of the initiator processor, and its connection to the right workflow.

4 Evaluation

The data importing pipeline is implemented with Apache NiFi and a set of RML Mappers. We ran an evaluation for 29 days between 23.06.2019 and 20.07.2019. We imported total nine named graphs: [11] mayrhofen, feratel; [21] seefeld, feratel; [31] serfaus-fiss-ladis, feratel; [12] mayrhofen, intermaps; [22] seefeld, intermaps; [32] serfaus-fiss-ladis, intermaps; [13] mayrhofen, outdooractive; [23] seefeld, outdooractive; [33] serfaus-fiss-ladis, outdooractive.

We calculated the average, median and standard deviation of both size and time elapsed for each import[13]. The entire graph (1.37M statements) is imported in approximately 26 minutes. This number is well within the daily frequency requirement. The largest graphs are provided from feratel, and they take

[12] https://nifi.apache.org/

[13] see the full statistics at : https://docs.google.com/spreadsheets/d/1C4aoBsN9p16LjPOLy33M9d8Mz0tFeUErrJDvRn

approximately 1ms per Triple (Table 2). The overall overhead of API access and authorization has a bigger impact than the actual generation of the data; therefore, the time needed for generating RDF triples is actually significantly below 1ms per triple[14]. Moreover, Apache NiFi allows parallelization and distribution, which we can adopt if necessary. We satisfy the requirement R-DI-scale.

The Apache NiFi instance provides a user interface for advanced settings and monitoring. Additionally, we provide a web interface for the creation of new imports. For mappings, we utilize a declarative mapping language. The whole pipeline can be managed without any programming knowledge. Therefore, R-DI-usability is satisfied. We attach provenance information to each imported statement. Additionally, Apache NiFi shows the provenance information at each processor and we provide a monitoring mechanism over the MindLab registration interface. This way, we satisfy the R-DI-prov.

Graph	Time/Tripple
[11]	1.28
[21]	1.15
[31]	1.05
[12]	6.40
[22]	21.73
[32]	4.11
[13]	5.81
[23]	7.30
[33]	10.99

Table 2. Data importer evaluation with 29 runs: Time/Triple (ms)

5 Related Work

Knowledge Graphs are being adopted for commercial usage internally to improve certain processes in a company and as a service to enable external applications (see [12] for some examples from big tech companies). We can classify the population process in four categories namely: (1) manual population in a closed system, (2) manual population by users/community, (3) automated population from (semi-)structured data sources, (4) automated population from unstructured data sources [5] [15]. Depending on the domain, one or a combination of some of the four approaches is used more dominantly. For example, in the health domain, it is common to extract knowledge from unstructured health records, whereas in the cultural heritage domain, several methods need to be combined due to the heterogeneity of the sources [10]. Many other examples of building Knowledge Graphs in enterprises can be found in works like [10, 14, 8].

A particularly related work to our Knowledge Graph construction pipeline comes from the automation industry [9]. Festo builds a Knowledge Graph to manage the technical configuration of their products in an efficient way. They utilize R2RML for transforming data from their RDBMS to RDF and apply OWL reasoning together with SWRL rules on the created data. Additionally, they control the dataflow with an Apache AirFlow based implementation. In our scenario, we map heterogeneous hierarchical sources to schema.org; therefore, we

[14] 0.04ms per triple with 500K triples. see the performance statistics of RocketRML in https://sumutcan.github.io/kgb-workshop-presentation/#/8/1

[15] We refer readers to Section 4 of Deliverable 2.1 of the MindLab project [2] for an overview of the population process of various open and proprietary Knowledge Graphs

use RML instead of R2RML. Moreover, we have an additional provenance model, since it is important to compartmentalize the data for different customers and applications.

6 Conclusion and Future Work

In this paper, we described our use case, namely the population of MindLab Knowledge Graph, that is generated based on heterogeneous semi-structured data (e.g., in JSON and XML). Our main focus is the knowledge generation pipeline that populates the MindLab Knowledge Graph. Currently, a big majority of our data comes from external APIs that provide data in JSON and XML format. We implemented a flexible and scalable architecture based on a high-performance RML mapper implementation and an open-source data flow management tool, Apache NiFi. We demonstrated that the developed tool satisfies the requirements of the MindLab project regarding provenance tracking, scalability, and usability.

The next steps will focus on assessing and improving the quality and on further enriching the knowledge of the MindLab Knowledge Graph. The quality improvement step typically contains cleaning tasks such as handling the detection and correction of errors, which may also be introduced by fusing duplicate instances (e.g., different set of opening hours for the same hotel). We want to enrich the knowledge graph by creating new relations between entities. This involves to add new relations between entities (e.g.,, equality statements, missing links, or pre-computed relations such as "inWalkingDistance"), but also new information not available in the data sources. The Knowledge Graph should contain not only facts but also the description of actions and services. For instance, given the hotel domain, we do not only want to represent and search for hotel information but also describe the actions related to hotels, such as booking a hotel room or some services offered by the hotel.

Acknowledgment

We would like to thank all participants of the MindLab project.

References

1. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. In: Proceedings of the 14th International Conference on World Wide Web. pp. 613–622. WWW '05, ACM, New York, NY, USA (2005). https://doi.org/10.1145/1060745.1060835
2. mindLab Consortium: D2.1 data import and management. Tech. rep., mindLab Project (2019)
3. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In: Proceedings of the 7th Workshop on Linked Data on the Web (Apr 2014), http://events.linkeddata.org/ldow2014/papers/ldow2014_paper_01.pdf

4. Fensel, D., Şimşek, U., Angele, K., Huaman, E., Kärle, E., Panasiuk, O., Toma, I., Umbrich, J., Wahler, A.: Knowledge Graphs - Methodology, Tools and Selected Use Cases. Springer (2020)
5. Färber, M., Bartscherer, F., Menne, C., Rettinger, A.: Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. Semantic Web **9**(1), 77–129 (2018), http://dblp.uni-trier.de/db/journals/semweb/semweb9.html#FarberBMR18
6. Guha, R.V., Brickley, D., Macbeth, S.: Schema.org: evolution of structured data on the web. Communications of the ACM **59**(2), 44–51 (2016). https://doi.org/10.1145/2844544
7. Hernández, D., Hogan, A., Krötzsch, M.: Reifying RDF: what works well with wikidata? In: Liebig, T., Fokoue, A. (eds.) Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems. CEUR Workshop Proceedings, vol. 1457, pp. 32–47. CEUR-WS.org (2015)
8. Hubauer, T., Lamparter, S., Haase, P., Herzig, D.M.: Use cases of the industrial knowledge graph at siemens. In: van Erp, M., Atre, M., López, V., Srinivas, K., Fortuna, C. (eds.) Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018. CEUR Workshop Proceedings, vol. 2180. CEUR-WS.org (2018), http://ceur-ws.org/Vol-2180/paper-86.pdf
9. Liebig, T., Maisenbacher, A., Opitz, M., Seyler, J.R., Sudra, G., Wissmann, J.: Building a knowledge graph for products and solutions in the automation industry. In: Proceedings of 1st Knowledge Graph Building Workshop co-located with 16th Extended Semantic Web Conference (ESWC), Portoroz, Slovenia, June 3, 2019. CEUR WS Proceedings (to appear) (2019), https://openreview.net/pdf?id=ByldIMlEDV
10. Monti, M., Perego, F., Zhao, Y., Vetere, G., Gomez-Perez, J.M., Alexopoulos, P., Nguyen, H., Webster, G., Villazon-Terrazas, B., Garcia-Santa, N., Pan, J.Z.: Success Stories, pp. 215–236. Springer International Publishing, Cham (2017), https://doi.org/10.1007/978-3-319-45654-6_8
11. Moreau, L., Groth, P., Cheney, J., Lebo, T., Miles, S.: The rationale of prov. Web Semantics: Science, Services and Agents on the World Wide Web **35**, 235 – 257 (2015), http://www.sciencedirect.com/science/article/pii/S1570826815000177
12. Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale knowledge graphs: Lessons and challenges. ACM Queue **17**(2), 20:48–20:75 (Apr 2019). https://doi.org/10.1145/3329781.3332266
13. Simsek, U., Kärle, E., Fensel, D.: Rocketrml - A nodejs implementation of a use-case specific RML mapper. In: Proceedings of 1st Knowledge Graph Building Workshop co-located with 16th Extended Semantic Web Conference (ESWC), Portoroz, Slovenia, June 3, 2019. vol. abs/1903.04969. CEUR WS Proceedings (to appear) (2019), http://arxiv.org/abs/1903.04969
14. Wood, D. (ed.): Linking Enterprise Data. Springer (2010), http://dblp.uni-trier.de/db/books/collections/W2010.html