A Method for Integrating Heterogeneous Data into a Knowledge Graph

Christoph Göpfert[1[0000-0001-6659-8947]], Sheeba Samuel[1[0000-0002-7981-8504]] and Martin Gaedke[1[0000-0002-6729-2912]]

[1] Technische Universität Chemnitz, 09111 Chemnitz, Germany
`{christoph.goepfert,sheeba.samuel,martin.gaedke}@informatik.tu-chemnitz.de`

Abstract. Knowledge graphs have gained popularity in many areas as a means of representing knowledge in a structured format. A large number of approaches for knowledge graph construction have been developed based on unstructured, semi-structured, or structured data sources. Many approaches focus on narrowly defined application areas or specific data source formats. In this paper, we describe a systematic method for knowledge graph construction from sources with heterogeneous data formats. The method comprises steps from ontology development to data acquisition and integration, mapping, and data refinement as well as the evolution of the constructed knowledge graph. We evaluate our approach using the case study of TUCgraph, a knowledge graph covering entities from the academic environment. This knowledge graph is actively being used within a research information system, demonstrating the practical applicability and effectiveness of our method.

Keywords: Knowledge Graph Development, Knowledge Graph Construction, Data Integration, Data Mapping, Data Evolution.

1 Introduction

Knowledge graphs (KG) are a powerful tool to represent knowledge in a structured manner. As such, KGs have been increasingly adopted across various areas [1–3]. A common challenge in KG construction (KGC) is the integration of heterogeneous data from distributed sources. Integration of these data into a uniform structure plays a vital role in comprehensive data analysis, knowledge discovery, and decision-making [1].

The integration of heterogeneous data poses several challenges: diversity in data formats, structures, and multiplicity of sources complicate data extraction, cleaning, and harmonization processes [4]. Failure to address these may result in incorrect representations in the KG. Therefore, there is a strong need for a robust, systematic approach to overcoming these challenges. While the need to update a KG when source data changes is considered by some KGC approaches, it is often limited to KG maintenance and one-directional updates [4], disregarding that changes may have occurred in the KG itself.

In this paper, we propose a six-step method for integrating heterogeneous data into a KG. We go beyond the sole consideration of maintenance aspects and address the evolution of data source(s) as well as the KG. In our case study, we demonstrate how the method was applied to construct a KG, which currently serves as the primary data source for a research information system.

2 Related Work

The development of ontologies, which provide the basis for KGs, is usually strongly dependent on the knowledge domain and format of the data source(s). Al-Zaidy and Giles [5] present a KGC approach to automatically extract entities and relations from scientific documents. Yet et al. [6] provide a survey that distinguishes generative KGC approaches, most of which assume unstructured text as source data. Kommineni et al. [7] illustrate an approach leveraging LLMs to support KGC tasks. However, a downside of automated approaches is that incorrect data may be generated, necessitating a subsequent review and refinement step. In addition, approaches to systematize KGC have been proposed. Asprino et al. [8] suggest an approach to transform heterogeneous data directly into the Resource Description Framework (RDF). However, their approach requires connectors dedicated to every source format. Tamašauskaitė and Groth [4] provide a literature review on KGC and propose a systematic six-step approach.

We present a systematic method that can be used to generate a KG from unstructured, semi-structured, or structured data sources. The method pays particular attention to the evolution of the KG and its data sources, an aspect that previous methods either neglected or only considered to a limited extent.

3 Method Description

Our method includes the steps: 1) ontology development, 2) data acquisition, 3) extraction and integration, 4) mapping, 5) refinement, and 6) update and evolution, which are based on previous methods such as [4]. However, we consider the ontology underlying the KG as a fundamental basis which should be in place in the data acquisition step. We also emphasize that a KG may be subject to change, i.e. evolving. Consequently, we extend the final step, which in previous approaches commonly involves only maintenance tasks, to account for the evolution of data source(s) and the KG itself.

3.1 Ontology Development

First, the underlying ontology of the KG must be developed. Entities and their relationships need to be described. This typically involves steps such as requirements specification, ontology design, construction, and evaluation [9]. For all identified entities, attributes and relations that are relevant to the selected domain or use case need to be identified. Next, the ontology can be modeled, manually or using an ontology editor. Based on the given entities, attributes and relationships, classes and properties are derived. Before defining new classes and properties, it should be checked if existing ontologies describe them and can be reused. Before defining new classes and properties, it should be checked if suitable ontologies exist that can be reused. Reusing ontologies reduces development effort and may enhance interoperability with established standards [8]. Once the ontology has been constructed, it should be evaluated based on the initially defined competency questions (cf. [9]). In addition, its quality should be evaluated. Should the evaluation yield unsatisfactory results, then the ontology should be improved, and the evaluation repeated, until a satisfactory result is achieved.

3.2 Data Acquisition

Data acquisition involves the identification and retrieval of data from one or multiple data sources. First, relevant data sources must be identified. Common techniques to obtain data include the utilization of APIs, which facilitate programmatic access to data provided by many online services and databases. To extract data from web pages, a popular method is web scraping. Another common data source is data dumps, which usually provide data in semi-structured formats, providing a convenient way to acquire a large amount of data at once. Once data sources have been identified and the data acquisition methods have been determined, the necessary infrastructure to collect and store data can be setup. Depending on the data volume and update frequency, this may include data pipelines, storage systems, and processing frameworks [10].

3.3 Extraction and Integration

Next, the acquired, raw data is converted into a structured format for integration. This step aims to ensure that the data is cleaned and curated, to enable an accurate mapping to the ontology in the next step. Extraction begins with identifying and isolating relevant entities, attributes, and relationships from the acquired data. Rule-based, dictionary-based, template-based approaches [11] may be used to automate this process fully or partially. In addition to entities, their describing attributes must also be extracted, as well as relations between entities. For this purpose, pattern-based or deep learning-based relation extraction approaches [12] can be utilized. The suitability of tools and techniques for extraction depends heavily on the format of the source data.
Once data is extracted, it needs to be cleaned. Data cleaning involves the correction or removal of incorrect, duplicate or incomplete data. This aims to recognize inconsistencies and errors in the data and, ideally, to correct them to improve data quality.

This is followed by data integration, in which data from multiple sources is harmonized and merged to create a unified data set. This may involve several challenges: different data sources may use different identifiers for one and the same entity, requiring a reconciliation of these identifiers to ensure consistency. In case of multiple data sources providing conflicting information about an entity, a conflict resolution strategy needs to be established. Finally, the result of this step is a unified dataset.

3.4 Mapping

The unified data must be aligned with the structure defined by the ontology. Entities must be assigned to appropriate classes; attributes, and relationships to corresponding data and object properties. For data type properties with a restricted value range, value type conversion is needed to ensure data integrity and consistency. This can be realized using the two-stage mapping process proposed by Asprino et al. [8]. First, mappings are defined to adhere to the structure of the ontology. Then, the mappings are processed to generate RDF. This is repeated – editing mappings and evaluating the generated RDF data against the input data to identify errors – until a satisfactory result is achieved.

3.5 Refinement

Refinement involves evaluating the accuracy, consistency, completeness and adherence to defined constraints, i.e. on data types, cardinality, properties, or further constraints set through additional rules, e.g. using SHACL [13]. In addition, a reasoning engine should be used to infer based on the rules defined in the ontology. Doing so may enrich the data by inference [12] and identify any violated constraints. For further approaches to KG refinement, we refer to the survey of Paulheim [14].

3.6 Update and Evolution

Changes in source data should be reflected in the KG to keep it up to date. This may be realized by subscribing to data update events (active update techniques), or by periodically checking each data source for changes [15]. Active updates are not always realizable, as it requires an event feed of recent changes. In such cases, an update frequency predictor can be employed [16]. To distinguish older from new data and to ensure only the latest information is used, a versioning system can be employed [4]. More complex situations may require merging changes from multiple sources, analogous as described in section 3.3. If the KG itself is subject to being modified, the update process becomes yet more complex, as data co-evolution (term as defined by Faisal et al. [17]) must be taken into account to integrate the updates to the data source(s) and the KG seamlessly.

Synchronization processes may also lead to errors, e.g. data inconsistencies, failed data updates, or incorrect merges. Therefore, the automation of synchronization with data sources requires careful monitoring and validation to detect errors. Automated processes must be able to handle situations where changes occur to a data source's schema that might disrupt data extraction and integration pipelines. Regular validation and plausibility checks may contribute to the early detection of such changes so that adjustments can be made to the extraction and mapping processes. Furthermore, over time it may be necessary to modify the ontology, refining or extending the ontology. This may require the migration of existing data in the KG to meet the requirements of the updated ontology. Consequently, assumptions made in the previous steps of extraction, processing, mapping, and validation steps may need to be reconsidered as well.

4 Case Study: TUCgraph

We applied the proposed method to construct TUCgraph, a KG currently used by the research information system (RIS) of Chemnitz University of Technology as the primary data source. At present (July 2024), TUCgraph contains 1,050,821 triples, aggregating and aligning data from multiple data sources. It entails entities such as university staff, organizational units, research projects, lab devices, events, and facilities.

Ontology Development TUCgraph is employed by a RIS based on the open-source platform VIVO [18]. As VIVO requires a predefined set of ontologies, these were (re-)used as the basis of TUCgraph to satisfy the requirement.

Data Acquisition The selected data sources include both internal university data sources and external API endpoints and websites. Whenever available, API endpoints were utilized. People and project data were enriched with data from external databases and websites of funding organizations. Data was partially obtained via web scraping due to the lack of a public API.

Extraction and Integration For each source format, a Python script was developed to acquire, validate and integrate data into an aligned structure. To disambiguate entities referred to by differing names, we used the entity linking approach of Zhao et al. [11]. The remaining ambiguities were addressed through manual review, however, some remained unresolvable due to missing contextual information.

Mapping The mapping step involved translating the aligned data onto the KG. In the case of TUCgraph, the target model is RDF. In this regard, entities and properties were mapped. Entities were mapped to their corresponding classes in the ontology and attributes and relations were mapped to their corresponding properties. We constructed SPARQL INSERT queries to insert the data directly into the KG.

Refinement We used a reasoner to find inconsistencies and datatype violations in the KG. The reasoner enriched the KG by inferring additional types of individuals, based on the rules of the ontology. In case of type violations, we either manually corrected the value, or, in case of uncorrectable, faulty data, removed it.

Update and Evolution We use a Python script for periodic updates of the KG. The script automates data acquisition, extraction, processing and integration, and mapping to generate SPARQL queries. As the KG is used by a RIS and can be modified by RIS users, a conflict resolution strategy is in place to prevent source data overwriting more recent user input. So far, a complex merge strategy was not needed in our case.

Despite its successful application, our method has limitations. The quality of source data in particular impacts the required effort, especially in step three where corrective measures are performed. Furthermore, in some cases, missing data could not be reliably inferred. This made data mapping impossible, effectively leading to data loss.

5 Conclusion

In this paper, we present a novel approach that considers KGC as a continuous process, addressing challenges from the co-evolution of source data and the KG, that were previously unaddressed. Our method fills this gap and offers guidance to KG engineers, contributing a practical solution for integrating multiple sources in heterogeneous formats into a unified KG. We demonstrate our method's applicability through TUCgraph, a KG developed in an academic setting. TUCgraph integrates a wide range of entities related to university operations. TUCgraph serves as the primary data source for a research information system, highlighting its utility.

Acknowledgments. This work was funded by German Research Foundation (DFG), TRR-386, TP INF, project number 514664767.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to this work.

References

1. Li, L. et al.: Real-world data medical knowledge graph: construction and applications. Artif. Intell. Med. 103, 101817 (2020).
2. Wang, C. et al.: Information extraction and knowledge graph construction from geoscience literature. Comput. Geosci. 112, 112–120 (2018).
3. Nayak, A. et al.: Knowledge Graph based Automated Generation of Test Cases in Software Engineering. In: Proceedings of the 7th ACM IKDD CoDS and 25th COMAD. pp. 289–295. ACM, New York, NY, USA (2020).
4. Tamašauskaitė, G., Groth, P.: Defining a Knowledge Graph Development Process Through a Systematic Review. ACM Trans Softw Eng Methodol. 32, 27:1-27:40 (2023).
5. Al-Zaidy, R.A., Giles, C.L.: Automatic Knowledge Base Construction from Scholarly Documents. In: Proceedings of the 2017 ACM Symposium on Document Engineering. pp. 149–152. Association for Computing Machinery, New York, NY, USA (2017).
6. Ye, H. et al.: Generative Knowledge Graph Construction: A Review, (2023).
7. Kommineni, V.K. et al.: From human experts to machines: An LLM supported approach to ontology and knowledge graph construction, (2024).
8. Asprino, L. et al.: Knowledge Graph Construction with a Façade: A Unified Method to Access Heterogeneous Data Sources on the Web. ACM Trans Internet Technol. 23, (2023).
9. Bravo, M. et al.: Methodology for ontology design and construction. Contad. Adm. (2019).
10. Lin, Z.-Q. et al.: Intelligent Development Environment and Software Knowledge Graph. J. Comput. Sci. Technol. 32, 242–249 (2017).
11. Zhao, Z. et al.: Architecture of Knowledge Graph Construction Techniques. IJPAM. 118, 1869–1883 (2018).
12. Yan, J. et al.: A retrospective of knowledge graphs. Front. Comput. Sci. 12, 55–74 (2018).
13. Dimitris Kontokostas, Holger Knublauch: Shapes Constraint Language (SHACL), (2017).
14. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web. 8, 489–508 (2016).
15. Wu, T. et al.: A Survey of Techniques for Constructing Chinese Knowledge Graphs and Their Applications. Sustainability. 10, 3245 (2018). https://doi.org/10.3390/su10093245.
16. Liang, J. et al.: How to Keep a Knowledge Base Synchronized with Its Encyclopedia Source. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. pp. 3749–3755. , Melbourne, Australia (2017).
17. Faisal, S. et al.: Co-evolution of RDF Datasets. In: Bozzon, A. et al. (eds.) Web Engineering. pp. 225–243. Springer International Publishing, Cham (2016).
18. Conlon, M. et al.: VIVO: a system for research discovery. J. Open Source Softw. 4, (2019).