An Operational Architecture for Knowledge Graph-Based Systems

Matthias Sesboüé, Nicolas Delestre, Jean-Philippe Kotowicz, Ali Khudiyev, Cecilia Zanni-Merk

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 207 (2022) 1667–1676

Procedia Computer Science

www.elsevier.com/locate/procedia

26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2022)

An Operational Architecture for Knowledge Graph-Based Systems

Matthias Sesboüé[a,*], Nicolas Delestre[a], Jean-Philippe Kotowicz[a], Ali Khudiyev[a], Cecilia Zanni-Merk[a]

[a]*INSA Rouen Normandie, LITIS, 76000 Rouen, France*

Abstract

Knowledge Graphs (KG) are gaining in popularity recently, notably since big tech giants announced they are using the technology. While the term is becoming popular, it is not new, and its ideas are even older. The research community has extensively studied knowledge Graphs in their various forms. Furthermore, the approach has been applied and proved valuable in many different applications. However, we found a lack of papers presenting the integration of KGs in a system regardless of the downstream application. We explore how KGs can fit in an overall information system independently from any specific use case, i.e., what we will consider knowledge consumption. We propose an architecture to understand better the KG roles within a system and how they can be integrated and implemented in a business context. We introduce each element of the latter architecture and discuss some candidate technology to implement them. Our work implements Knowledge Graph-Based Systems considering the constraints of a small to medium-sized enterprise.

Keywords: Knowledge Graph ; Ontology ; Knowledge-Based System ; Semantic

1. Introduction

Knowledge Graphs (KG) are gaining in popularity recently, and in particular, since big tech giants announced they are using the technology, e.g., Google Knowledge Graph in 2012. In 2020, Gartner placed KGs at the peak of inflated expectations in their Hype Cycle for artificial intelligence. However, the term is not new, and its ideas are even older. Indeed, in 1987, R.R. Bakker titled his PhD thesis “Knowledge Graphs: Representation and Structuring of Scientific Knowledge” [2]. Among different communities, different names have been used. They have been applied in multiple use cases and have proven a critical tool in various domains. In particular, in life science and medicine. While the research community has extensively studied knowledge Graphs in their various forms, we found a lack of papers presenting the integration of KGs in a system regardless of the downstream application. Most papers either mention they are using a KG to support their use case, or are focusing on its creation process, e.g., [15]

* Corresponding author. Tel.: +33 6 47 03 99 84
E-mail address: matthias.sesboue@insa-rouen.fr

10.1016/j.procs.2022.09.224

This paper studies Knowledge Graph-Based Systems (KGBSs) and their implementation. Despite having great promises, we recognise that their implementation has always been a challenge, particularly for small to medium-sized industries with limited resources. We explore how KGs can fit in an overall information system regardless of any specific use case, i.e., what we will consider knowledge consumption. To better understand the KG roles within a system and how they can be integrated and implemented in a business context, we propose an architecture for such a system. In the remaining of this paper, we first review related works, retracing the history of the core ideas behind KGs. We then present a functional architecture for KGBSs and discuss candidate technologies to implement each architecture element. We conclude by considering a Knowledge Graph-based approach to Information Retrieval to illustrate our arguments.

2. Related work

This section reviews related works and introduces the Knowledge Graph ideas' origins. We first contextualize the notion of Knowledge Graphs retracing its core ideas and then recognize a gap between academia and industry notions of KGs.

2.1. Knowledge Graphs genesis

Knowledge Graphs achieve an early vision in computing of creating intelligent systems that integrate knowledge and data on a large scale [21]. To better understand the core ideas behind this technology, it is essential to consider its genesis. In [12], Gutierrez and Sequada provide details on the historical context of Knowledge Graphs as we know them today. They select core ideas and group them into five themes corresponding to periods from the advent of computing in its modern sense in the 1950s to the present day. They describe each period following two axes, data and knowledge, and how they relate. Knowledge Graphs as we know them today result from the convergence of multiple scientific and societal facts. In the following, we summarize the history of Knowledge Graphs using the terms coined by Gutierrez et al. For further details, we recommend [12].

In the 1950s, the advent of the digital age is marked by the spread of digital computers. There was a need to understand natural language and other human knowledge representations to automate reasoning. Original ideas of KGs were introduced in what was known as Semantic Networks [4]. The notion of networks stressed the potential of graphical representations in general as a tool for abstraction. The data and knowledge foundations era in the 1970s witnessed a much broader adoption of computing in the industry. Research on knowledge representation focused on the meaning of data. Semantic Networks were under critique because of their weak logical foundations. Formal logic was considered a tool to build a formal framework to support Semantic Networks. Sowa proposed a formalization with conceptual structures in 1984 [23]. Still, in the 1980s, the need for a trade-off between the expressive power of logical languages and the computational complexity of reasoning tasks arose. It is the coming age of data and knowledge integration. Expert systems were at the centre of the AI hype, and the research community recognized the knowledge acquisition bottleneck. The last decade of the 19th century witnessed the emergence of the Web and the digitization of almost all aspects of society. Tim Berners Lee introduced the Semantic Web project [3] intending to converge technologies such as knowledge representation, ontologies, logic, databases, and information retrieval on the Web. Gruber defined Ontologies as a "shared and formal specification of a conceptualization". In the 2000s, the computing power and cheaper access to resources eventually resulted in the storage and processing of vast amounts of data. During the Big Data era, symbolic approaches to knowledge processing aroused less interest, and we witnessed a spike in the adoption of nonsymbolic statistical methods.

Nowadays, we have access to vast amounts of data and knowledge gathered during decades of industry leveraging computing power. However, we also lose track of the meaning behind our data as we tightly couple them with applications. The industry is now looking to restore meaning to its data assets, and we see terms such as "data-centric" and "semantic layer" emerging. The research community has begun to study how symbolic and nonsymbolic approaches could merge.

2.2. Knowledge Graphs

As mentioned previously, Knowledge Graphs are a convergence of old ideas and technological advances. In [21], the authors define it as representing "a collection of real-world concepts (i.e., nodes) and relationships (i.e., edges) in the form of a graph used to link and integrate data coming from diverse sources". They further break down the term "Knowledge Graph". They define "knowledge" as representing the meaning embedded in the relations between and within data and "Graph" as the data structure enabling data integration from heterogeneous sources. Even though the core ideas stay the same, some differences arise in the implementations and the usage. Based on our experience with both academic and industry-focused communities, we loosely distinguish between the academic and industry interpretation of a KG.

Academics, especially the logic and semantic web communities, use "ontology". However, although the scientific articles constantly refer to the same papers ([10], [11], [13], [24]), the definitions are vague. Each has its strengths and weaknesses and leaves space for a wide range of interpretations [14]. One commonality at the implementation level is using languages with solid logic foundations, such as OWL, which the W3C standardizes. We observe that the logical foundations of ontologies as defined by academics are not well understood on the industry side. A Knowledge Graph is occasionally defined as an ontology with the data. This definition considers ontology as a schema and contradicts the Semantic Web notion of ontology and its Description Logic-based implementation, which distinguishes the Assertions Box (A-Box), and the Terminology Box (T-Box).

Additionally, we distinguish two notions of semantics as the industry adopted the term. There are at least three distinct ways to consider these two approaches to representing meaning, i.e., symbolic versus nonsymbolic, knowledge versus data-driven, and top-down versus bottom-up. The nonsymbolic and data-driven semantic known as embeddings are numerical vectors attempting to represent meaning extracted from loads of examples. We often encounter the latter interpretation in the industry. In contrast, symbolic and knowledge-driven semantics develop a logic-based meaning representation. The latter is the focus of our work, even though we agree that both approaches should work together in a successful system.

In the following, we intentionally use the term KG to keep our proposition generic and not commit to any specific implementation. Both academia and industry are now adopting the KG keyword to loosely designate systems that integrate data with some structure of graphs.

3. An architecture for Knowledge Graph-Based Systems

As shown in the previous sections, the scientific community has studied the various aspect of KG-based systems since the first computers showed value in the industry. We do not need to stress the theoretical opportunities modelling knowledge in a machine-interpretable format can bring to an industry. However, the main challenges arise when implementing and using KGBSs. This section discusses a general architecture for a knowledge-based system, regardless of the downstream application. We describe the individual components in figure 1 and give some examples.

Figure 1 presents an operational architecture for KGBSs. Round boxes refer to activities and their sub-activities. Squared boxes represent containers with their content clustered for ease of understanding. Arrows illustrate data flows. While we can easily translate some activities to a program, some also denote a human thought process. In the following, we discuss each element depicted in figure 1.

3.1. Knowledge acquisition

The literature often refers to knowledge acquisition as one activity, e.g., in [17]. This paper breaks down the latter process into knowledge elicitation, knowledge extraction, and consumption. We keep in mind that before modelling any knowledge, we need to scope our project and define the concepts and relations of interest. Knowledge elicitation aims at extracting concepts required to answer the business questions. In IR use cases, concepts are hidden in a corpus of documents or user queries. The data profiler computes and displays various statistics driven by the exploration of the knowledge scientist [7]. The latter analysis highlights a business question or, in our IR use case, a user query concept, e.g., an identifier.

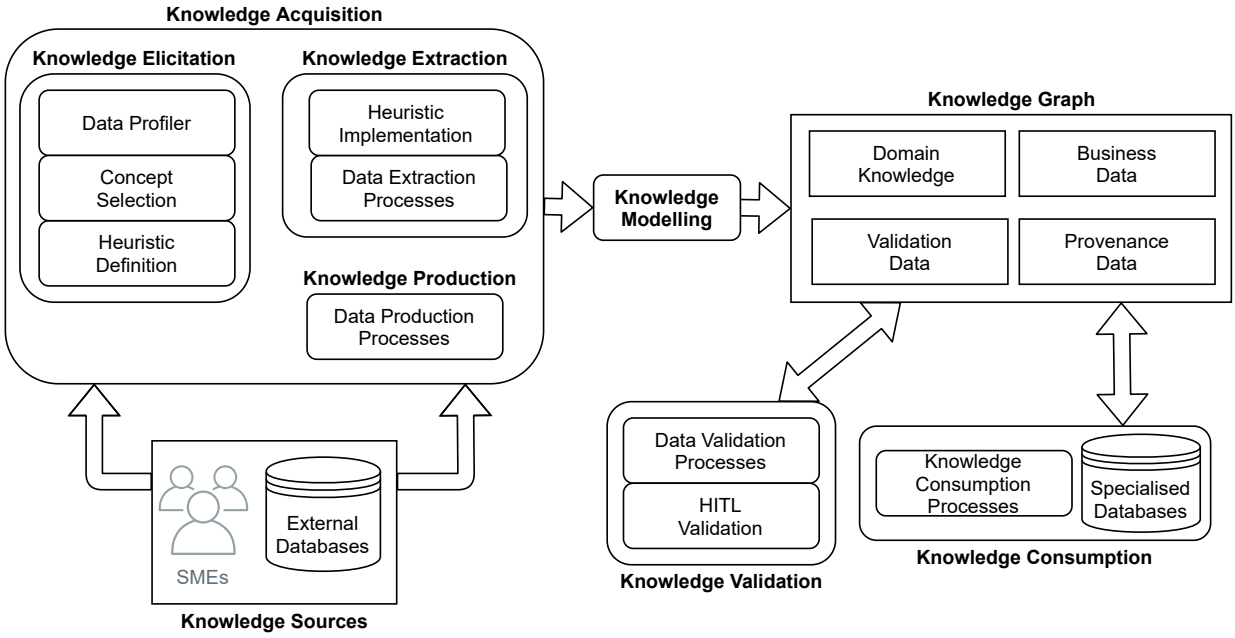

Fig. 1. Knowledge Graph-Based System architecture. (*SMEs*: Subject Matter Experts; *HITL*: Human In The Loop)
Round boxes refer to activities and their sub-activities. Squared boxes represent containers with their content clustered for ease of understanding. Arrows illustrate data flows between activities and components.

After selecting a concept, we define a heuristic to extract instances from documents or enterprise resources. An example of a heuristic is that phone numbers are sequences of ten digits with potentially an optional double zero or plus sign and a two digits country code before. We extract concept instances after implementing the heuristic, e.g., with a regular expression. These processes are similar to Extract Transform and Load (ETL) ones. We distinguish between the latter extraction process and the production one, which applies transformations, so the data fit the KG structure. Assuming we use an object-oriented programming language for knowledge extraction and decide to represent our KG with the data representation paradigm of RDF triples, the knowledge extraction phase will construct objects containing all the extracted pieces of information. The knowledge production subprocess will build RDF triples based on the objects created in the previous phase.

3.2. Knowledge modelling

It makes sense to read the diagram from left to right. However, the components' positions do not intend to reflect any order in the activities. Indeed the method we will discuss in the following illustrates a cyclic iterative process.

The main container is the Knowledge Graph. It has the critical role of holding the single source of truth. The graph groups multiple kinds of knowledge we break down into four components. The sole goal of the knowledge acquisition process is to model and relate core concepts, both among them and with the business data. The other parts of the KG are necessary metadata for lineage and maintenance. They are of utmost importance to ensure the KG's organic evolution and trustworthiness. The domain knowledge represents concepts and relations describing the domain of interest. For instance, in a mechanical engineering application, domain knowledge could be standard notations, specific dimensions and terms. The most common examples are modelling units, dimensions, time and domain-specific terms. Some work proposes to break down this domain knowledge into categories such as top-level, domain, task and application knowledge [11]. Those classifications of knowledge might be pertinent, particularly for documentation and ease of understanding purposes. However, the separation between the categories quickly becomes fuzzy. From an implementation point of view, their need is less critical since, as we will discuss in the next section, we represent knowledge in one single format. What remains essential is to share a common understanding of the

knowledge stored. Business data groups all the application-specific information. It can be data ranging from employee details for the HR department to marketing documents targeting potential customers.

Nowadays, metadata is the kind of data we are missing. We store loads of information and produce even more from them, but we lack metadata to keep track of their meaning and origin. The domain knowledge adds context and meaning to business data. However, to trust the information we extract from the latter and make informed business decisions, we need to know precisely where each piece of data comes from and what processes produce it. Keeping track of such provenance knowledge is known as data lineage. Validation data describe the shape of data, e.g., an employee must have a social security number and a ten-digit phone number. Such metadata is necessary to ensure stored knowledge quality over time and ease its maintenance. It is not to be confused with domain knowledge, which describes data semantics, e.g., a person employed by a company is an employee of the latter, and a social security number is an identifier that uniquely identifies her or him.

3.3. Knowledge validation and consumption

In the previous paragraph, we introduce data for knowledge validation. They are data encoded using a specific language. Data validation processes are software designed to understand the latter language and process it on the knowledge graph to validate the semantic consistency and the shape of the data. It is essential to distinguish between shape data validation and semantic validation, which concentrates on logical consistency. The latter is more challenging and requires domain experts to validate the encoded logic. Those processes enable validating data automatically. Nevertheless, some validations require to include Humans In The Loop (HITL).

In figure 1, we abstract any specific use cases in the knowledge consumption activity. Typically some machine learning processes infer new knowledge, which we then add to the KG. Alternatively, some part of the graph is loaded in a particular database for performance reasons when an application needs to consume the knowledge. The critical point here is that the KG is the single source of truth, and as such, the application database is responsible for staying in sync and up to date with it. It should not be the other way around, as it is often the case [1] [18].

3.4. Knowledge sources

Methodologies for knowledge acquisition consider an ideal setting with access to Subject Matter Experts (SMEs) [15]. Experts are rarely available in practice. Indeed, it would be ideal to have unlimited access to a group of domain experts and translate the knowledge they are sharing into a machine-interpretable model. Unfortunately, very few companies have the resources to invest enough to allocate SMEs to a knowledge-sharing activity fully. The lack of access to experts is one of the main challenges for knowledge acquisition. In figure 1, knowledge sources include SMEs for completeness.

Most knowledge is hidden in companies' databases, schemas, documents and codebases. Each employee is an expert in one aspect of the company activity. They encode their knowledge in various forms. For example, the developer introduces knowledge in the code s·he writes. The problem is not the access to knowledge but rather the multiple forms it has and the vast amount we need to process.

4. Discussing some candidate technologies

In the previous section, we propose an architecture for a KG-based system. We discuss each element from an abstract point of view without considering any implementation. Our work wishes to path the way for a better understanding of KG-based systems implementation. Hence, in the following, we discuss candidate technologies.

[1] `http://datacentricmanifesto.org/` (Accessed on May 24, 2022)

4.1. Different Knowledge Graph paradigms

Knowledge graphs play a role in a wide range of applications. Depending upon the use case, its implementation varies. Before addressing our metadata modelling use case, we argue that we can roughly distinguish between two kinds of graph usage in the industry.

When the problem is a typical graph one, e.g., centrality or shortest path, or considered a graph problem for solving purposes, a graph database is used to store and manipulate the data. This graph database is said to store the KG. In figure 1, it would correspond to a knowledge consumption activity discussed in section 3.3. Typical use cases are any analytics performed on networks of any kind. The second use of graphs for modelling is to represent metadata. This use case corresponds to our notion of knowledge graph since it considers the knowledge gathered and formalised as metadata for a downstream application, e.g., enhancing a CAD model retrieval system. It responds to a data management problem and is typical of the academic notion of logic-based semantics.

When implementing a KG, many solutions exist. From the initial business constraints and needs originate the final implementation details. There are three main kinds of graph representations implemented in the database industry. The Labelled Property Graph (LPG) model tends to be the most widely spread modelling choice, probably because of its low barrier to adoption. This graph structure allows any amount of property attached to both the nodes and the relations, which is the most natural way of representing real-world entities in a graph. It is a typical modelling choice for analytics problems relying on graph algorithms. The hypergraph model distinguishes itself from the LPG in that it allows a relationship to connect any number of nodes. It is handy for data involving multiple many-to-many relationships. Some argue that although, in theory, hypergraphs produce accurate, information-rich models, in practice, it is pretty easy to miss some detail while modelling [20]. The RDF triples model is the simplest graph model relying on a representation composed of a set of triples, i.e., two nodes and a link. Its simplicity is its strength as well as its weakness. Indeed, this model, initially designed as a framework for expressing information about resources on the web, forces some complex triple structures to represent data easy to model in other graph models. An example is Wikidata [25], which has an internal structure of LPG but is also available as RDF triples with complex structure. However, RDF is at the root of the W3C semantic standards, making the data modelling independent of the database implementation. Note that ongoing community efforts aim at standardising the LPG model[2] and merging the latter with RDF[3]. It will probably eventually become the RDF-star [4] standard.

We advocate for the latter framework for various reasons, mainly because we are tackling a metadata problem. The W3C semantic web standards were designed for modelling and exchanging metadata. Many technologies have been developed on top of RDF and implemented in large scale projects. Hence, we do not wish to reinvent the wheel. This framework provides all the required languages, and the vast community of users have developed tools to manipulate them. RDF also comes built-in with a unique identifiers scheme. We can exploit identifiers globally unique, internally and externally, to explore and integrate external knowledge bases such as the DBpedia databases or Wikidata [25], to name the most well-known ones.

4.2. Knowledge Graphs implementation

Many commercial solutions exist to store the triples since the graph database market is still in its early days. There is a constantly changing landscape of newcomers [21], though we are interested in open source solutions.

There exist two main opensource frameworks for triple storage and manipulation, Oracle RDF4J and Apache Jena. They are complete frameworks with built-in components for storing, querying, reasoning, loading, and exporting RDF triples. They are also flexible enough to integrate other solutions, such as commercial ones, to replace specific elements. The latter fact demonstrates the advantage of using a standardised representation such as RDF. Using this format, we are only committing to the standard and not to any software vendor. For instance, the storage layer could be any database, ranging from RDBMS to document-oriented ones and passing by native graph storage.

[2] `https://www.w3.org/community/propertygraphs/` (Accessed on May 24, 2022)
[3] `https://w3c.github.io/rdf-star/cg-spec/2021-12-17.html` (Accessed on May 24, 2022)
[4] `https://w3c.github.io/rdf-star/cg-spec/2021-12-17.html` (Accessed on May 24, 2022)

4.3. Knowledge acquisition

In section 3.1, we break down the knowledge acquisition process into knowledge elicitation, extraction and production. Knowledge elicitation can involve technological tools, though it is primarily human work. Fletcher et al. define the role of the knowledge scientist in [7]. As detailed in the latter paper, knowledge science is technical, and people work. Tools can help but not replace the knowledge scientist. In particular, technological tools could help engage the domain experts in the knowledge acquisition process.

As already mentioned in section 3.4, having sufficient access to domain experts is demanding and unrealistic for many businesses. However, most knowledge is hidden in various formats and mainly in text. Hence, though out of the scope of our research, topics related to Natural Language Processing are of interest. In particular, we pay attention to the emerging area of Technical Language Processing [5] and wish to apply some of the work from this community. We do not focus on advancing the Natural Language Processing (NLP) research but instead use some results. We consider a python framework such as Spacy.

4.4. Knowledge modelling

In section 3.2, we break down the KG into four components, but we do not mention any standard languages. We wish to model domain knowledge in OWL eventually. However, we recognize that OWL's complexity and logical foundations make it hard to adopt from an industry point of view. The open-world assumption is often hard to grasp, and most employees are familiar with the object-oriented programming principles, which leads to lousy modelling practices and misunderstanding of the inferences.

We notice that most knowledge graph projects leveraging inference are limited to the hierarchical ones, i.e., the transitivity of hierarchical relations. RDFS or SKOS languages are more accessible and often sufficient to initiate a knowledge modelling project. Within a company, there often already exists, in various forms, classifications of the business concepts. The first step is to standardize them before enriching them. When the need for more complex descriptions arise, we can move from SKOS to OWL. The SKOS standard discusses the link between SKOS concepts and OWL classes [5]. Note that not all concepts modelled in SKOS have to become an OWL class, i.e., not all concept needs an advanced description in OWL. Both modelling languages can work alongside. RDFS, SKOS and OWL together enable us to model domain knowledge and business data.

It is often tempting to model an entire document as precisely as possible. We might need such a level of detail in the future, but most certainly not at the beginning. First, having a URI uniquely identifying a physical business document is enough to build its related knowledge in an iterative process.

4.5. Knowledge provenance

The W3C (World Wide Web Consortium) Provenance Working Group's definition of provenance is a record that describes the people, institutions, entities, and activities involved in producing, influencing or delivering a piece of data or a thing [16]. The PROV standard includes PROV-O, an OWL2 ontology with all the building blocks to describe provenance information. Among the concepts, notice the notion of activity, which concerns any processes involved in data production, such as extracted data from a Relational DataBase Management System (RDBMS) or generating triples based on data extracted from text.

To be as complete as possible in the expression of data provenance, we consider other languages with mapping to the RDF data model. RML [6] and its standardized extension for RDBMS, R2RML, let us state explicitly mappings between data stored in any database and the type of RDF term that needs to be generated. For triples directly generated from text analysis processes, we are exploring a recent effort to formalize Ontology Design Patterns[8] using RDF, namely Reasonable Ontology Templates (OTTR)[22]. The latter template language lets us explicitly describe macros for triple creation.

[5] `https://www.w3.org/TR/skos-primer/#secskosowl` (Accessed on May 24, 2022)

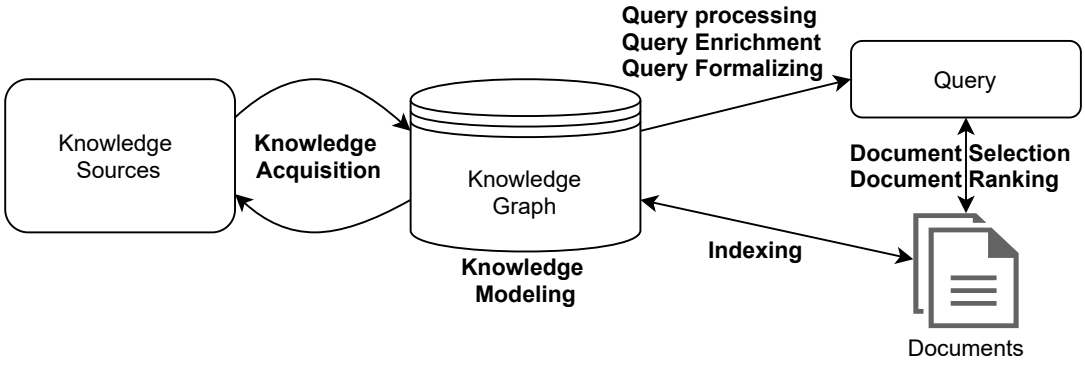

Fig. 2. KG-based IR system overview.

4.6. Knowledge validation

To maintain a KGBS, we need to reconsider an old idea [1] we see coming back in the data management industry, namely applying technics we use in software development to data. Some advocate for treating data as a product [19] and work on adapting continuous integration and development ideas to data. We see terms such as "dataOps" [6] emerging on the data analytics side. On the knowledge representation side, the term "SemOps" [7], the contraction of semantics and DevOps, is emerging.

OTTR templates contribute to the KG quality by consistently generating triples. To apply a continuous development and integration approach to data production, we need to validate the structure of data generated by diverse processes. It is also critical for data consumption as applications might rely on this latter structure to function. The industry has been working on the latter issue, and a recent W3C effort led to a standard, the SHape Constraint Language (SHACL). We can also mention the Shape Expression (ShEx) language, which has a goal similar to SHACL, although both technologies solve the problem from different perspectives and formalisms [9].

It would be idealistic to believe that the knowledge validation processes can all be performed automatically. It is critical to involve and engage human experts to ensure quality maintenance. The study of solutions to validate RDF data structure involving humans is out of the scope of this paper. However, some related topics are ontology verbalization, natural language generation (NLG) and chatbots. We could imagine generating natural language sentences from KG content, e.g., combining technics employed in ontology verbalization and NLG and using the result to engage experts in the validation process through a chatbot conversation.

Section 3.3 mentions the difference between data and semantic validation. We address the former with either SHACL or ShEx and a suitable processor. We consider the latter by using the OWL2 languages alongside a reasoner to spot any logical inconsistencies. Note that RDFS is a simple ontology language and does not contain the logical notion of negation. As such, RDFS reasoning cannot create or reveal inconsistencies. Semantic validation requires the expressivity of OWL. However, some of this validation can also be performed by employing rules encoded in SWRL or SPARQL queries.

5. An Information Retrieval use case

To illustrate our argument, we consider a Knowledge Graph-based approach to Information Retrieval. Figure 2 presents an overview of the main components and processes involved in a Knowledge Graph-Based IR system. The central element is the Knowledge Graph (KG). The knowledge is acquired and modelled to feed the KG. A user

[6] `https://medium.com/data-ops/dataops-is-not-just-devops-for-data-6e03083157b7` (Accessed on May 24, 2022)
[7] `https://www.semanticarts.com/the-data-centric-revolution-the-role-of-semops-part-1/` (Accessed on May 24, 2022)

expresses her or his needs with a query. The IR system process the latter, possibly involving multiple tasks. The resulting query is used to select and rank documents. Note that the latter is indexed based on the KG concepts.

In figure 1, We abstract the specific application of IR in the knowledge consumption component presented in section 3.3. For instance, we might extract the domain and business knowledge parts of the KG and store them in a specialized database to index documents based on them. Such a system could be an Elasticsearch engine which integrates the document indexing, selection, and ranking. This engine also includes solutions for textual query processing. We can use a Named Entity Recognition (NER) system to extract the concepts expressed in the text if considering the textual user query. The latter concepts enable us to formalize the query. We can then use a SPARQL engine to query over the domain knowledge and enrich the user query. To enrich the user query with the most accurate concepts, the KG must be as complete and clean as possible. The validation data introduced in section 3.3, and the processes and technologies mentioned in section 4.6 are critical in this endeavour.

Knowledge is not static, so should be the KG. The knowledge acquisition concepts depicted in section 3.1 and their candidate implementations proposed in section 4.3 are essential to any KG-based system, regardless of the knowledge consumption use cases. In figure 2, the knowledge sources are the same as discussed in sections 3.4 and 4.3. They could be anything ranging from the outcome of brainstorming sessions with domain experts to a piece of code encoding a business logic. Each source might require its specific knowledge acquisition processes.

6. Conclusion

This paper proposes an architecture for KG-based systems to better understand the KG role and integration within a system. We also discussed candidate technologies for each of our architecture components. In particular, we advocate for RDF, the W3C standard, and the multiple languages and tools built on top of it. We motivate its use and present potential languages.

Our work focuses on providing an approach to building a KG-based system tractable for small to medium-size enterprises. We consider existing research, focus on applying methods to an industrial environment, and explore candidate solutions considering business constraints. The architecture is a first step. However, we recognise the knowledge acquisition bottleneck and are investigating solutions. For future work, we read figure 1 presenting the architecture proposed in this paper from left to right and investigate an iterative approach to Knowledge acquisition and implementation derived from [21].

This work fits in with trends emerging in the data management industry. Our methods follow the data-centric principles[8] and treat data as a product by applying development and maintenance techniques adapted from the software industry. Related keywords are “dataOps”[9], “SemOps”[10], and “knowledge scientist”, as defined in [7]. Finally, during our research, we acknowledge that data management is a technological problem and a people problem [21]. Hence, our work can be successful only if we consider people alongside data management.

References

[1] Alonso, F., Juristo, N., Maté, J., Pazos, J., 1996. Software engineering and knowledge engineering: Towards a common life cycle. Journal of Systems and Software 33, 65–79. doi:10.1016/0164-1212(95)00104-2.

[2] Bakker, R.R., 1987. Knowledge Graphs: representation and structuring of scientific knowledge.

[3] Berners-Lee, T., 1998. Semantic web road map. Online. URL: https://www.w3.org/DesignIssues/Semantic.html. accessed on 04/04/2022.

[4] Brachman, R.J., 1979. On the epistemological status of semantic networks**prepared in part at bolt beranek and newman inc. under contracts sponsored by the defense advance research projects agency and the office of naval research. the views and conclusions stated are those of the author and should not be interpreted as necessarily representing the official policies, either express or implied, of the defense advanced research projects agency or the u.s. government., in: FINDLER, N.V. (Ed.), Associative Networks. Academic Press, pp. 3–50. URL: https://www.sciencedirect.com/science/article/pii/B9780122563805500074, doi:https://doi.org/10.1016/B978-0-12-256380-5.50007-4.

[8] http://www.datacentricmanifesto.org/ (Accessed on May 24, 2022)

[9] https://medium.com/data-ops/dataops-is-not-just-devops-for-data-6e03083157b7 (Accessed on May 24, 2022)

[10] https://www.semanticarts.com/the-data-centric-revolution-the-role-of-semops-part-1/ (Accessed on May 24, 2022)

[5] Brundage, M.P., Sexton, T., Hodkiewicz, M., Dima, A., Lukens, S., 2021. Technical language processing: Unlocking maintenance knowledge. Manufacturing Letters 27, 42–46. URL: https://www.sciencedirect.com/science/article/pii/S2213846320301668, doi:https://doi.org/10.1016/j.mfglet.2020.11.001.

[6] Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R., 2014. RML: a generic language for integrated RDF mappings of heterogeneous data, in: Bizer, C., Heath, T., Auer, S., Berners-Lee, T. (Eds.), Proceedings of the 7th Workshop on Linked Data on the Web. URL: http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[7] Fletcher, G., Groth, P., Sequeda, J., 2020. Knowledge scientists: Unlocking the data-driven organization. arXiv:2004.07917.

[8] Gangemi, A., Presutti, V., 2009. Ontology design patterns, in: Handbook on Ontologies. Springer Berlin Heidelberg, pp. 221–243. doi:10.1007/978-3-540-92673-3_10.

[9] Gayo, J.E.L., Prud'hommeaux, E., Boneva, I., 2017. Validating RDF Data. Morgan & Claypool Publishers. URL: https://www.ebook.de/de/product/30338735/jose_emilio_labra_gayo_eric_prud_hommeaux_iovka_boneva_validating_rdf_data.html.

[10] Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition 5, 199–220. URL: https://www.sciencedirect.com/science/article/pii/S1042814383710083, doi:https://doi.org/10.1006/knac.1993.1008.

[11] Guarino, N., 1998. Formal ontology and information systems.

[12] Gutierrez, C., Sequeda, J.F., 2021. Knowledge graphs. Communications of the ACM 64, 96–104. doi:10.1145/3418294.

[13] Horrocks, I., Patel-Schneider, P.F., van Harmelen, F., 2003. From shiq and rdf to owl: the making of a web ontology language. Journal of Web Semantics 1, 7–26. URL: https://www.sciencedirect.com/science/article/pii/S1570826803000027, doi:https://doi.org/10.1016/j.websem.2003.07.001.

[14] Keet, C.M., 2020. An Introduction to Ontology Engineering. https://people.cs.uct.ac.za/m̃keet/OEbook.

[15] Kendall, E.F., McGuinness, D.L., 2019. Ontology engineering. Synthesis Lectures on the Semantic Web: Theory and Technology 9, i–102. doi:10.2200/s00834ed1v01y201802wbe018.

[16] Luc Moreau, P.G., 2013. Provenance. Morgan & Claypool Publishers. URL: https://www.ebook.de/de/product/21525126/luc_moreau_paul_groth_provenance.html.

[17] Mariano Ferndndez, Asuncion Gomez-Perez, N.J., 1997. Methontology: From ontological art towards ontological engineering. AAAI Technical Report SS-97-06 URL: https://www.aaai.org/Papers/Symposia/Spring/1997/SS-97-06/SS97-06-005.pdf.

[18] Mccomb, D., 2018. Software Wasteland: How the Application-Centric Mindset is Hobbling our Enterprises. TECHNICS PUBN LLC. URL: https://www.ebook.de/de/product/31799401/dave_mccomb_software_wasteland_how_the_application_centric_mindset_is_hobbling_our_enterprises.html.

[19] Patil, D.J., 2012. Data jujitsu : the art of turning data into product. O'Reilly, Sebastopol, CA.

[20] Robinson, I., 2015. Graph databases: new opportunities for connected data. O'Reilly, Sebastopol, CA.

[21] Sequeda, J., Lassila, O., 2021. Designing and building enterprise knowledge graphs. Synthesis Lectures on Data, Semantics, and Knowledge 11, 1–165. doi:10.2200/s01105ed1v01y202105dsk020.

[22] Skjæveland, M.G., Lupp, D.P., Karlsen, L.H., Forssell, H., 2018. Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates, in: Lecture Notes in Computer Science. Springer International Publishing, pp. 477–494. doi:10.1007/978-3-030-00671-6_28.

[23] Sowa, J.F., 1984. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley Longman Publishing Co., Inc., USA.

[24] Studer, R., Benjamins, V., Fensel, D., 1998. Knowledge engineering: Principles and methods. Data & Knowledge Engineering 25, 161–197. URL: https://www.sciencedirect.com/science/article/pii/S0169023X97000566, doi:https://doi.org/10.1016/S0169-023X(97)00056-6.

[25] Vrandečić, D., Krötzsch, M., 2014. Wikidata: A free collaborative knowledgebase. Commun. ACM 57, 78–85. URL: https://doi.org/10.1145/2629489, doi:10.1145/2629489.