Web Semantics: Science, Services and Agents on the World Wide Web 84 (2025) 100846

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Process Knowledge Graphs (PKG): Towards *unpacking* and *repacking* AI applications

Enrico Daga[1]

Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, United Kingdom

ARTICLE INFO

Keywords:
Knowledge graphs
Prompt engineering
Data science pipelines
Data pipelines documentation
Data pipelines design

ABSTRACT

In the past years, a new generation of systems has emerged, which apply recent advances in generative Artificial Intelligence (AI) in combination with traditional technologies. Specifically, generative AI is being delegated tasks in natural language or vision understanding within complex hybrid architectures that also include databases, procedural code, and interfaces. Process Knowledge Graphs (PKG) have a long-standing tradition within symbolic AI research. On the one hand, PKGs can play an important role in describing complex, hybrid applications, thus opening the way for addressing fundamental challenges such as explaining and documenting such systems (unpacking). On the other hand, by organising complex processes in simpler building blocks, PKGs can potentially increase accuracy and control over such systems (repacking). In this position paper, we discuss opportunities and challenges of PGRs and their potential role towards a more robust and principled design of AI applications.

1. Introduction

The advent of large-scale generative Artificial Intelligence (AI) models has impacted the design of applications in many critical domains [1]. A new generation of systems has emerged, integrating generative AI with traditional technologies. Generative AI is being delegated tasks in natural language or vision understanding to automate operations that previously needed substantial human intervention. Empirical research in prompt engineering makes generative AI ready to perform many knowledge-intensive tasks, including automating complex workflows involving critical decisions in domains such as healthcare [2] and automotive [3]. Prompt engineering patterns' catalogues are emerging, to support the design and reuse of gen-AI tasks [4]. In real-world applications, generative AI is deployed as part of complex hybrid architectures that also include databases, procedural code, and interfaces. Toolkits are emerging to facilitate the structuring of complex interactions, such as Langchain [5] or Metagpt [6].

Knowledge graphs are important in allowing open-ended data sources to be seamlessly integrated into machine learning and LLM-powered, hybrid AI applications [7,8]. Knowledge Graphs can complement limitations of sub-symbolic methods such as support explainability [9]. In addition, the semantic web community is looking into how LLM can automate ontology engineering tasks [10,11].

In this position paper, instead, we focus on Process Knowledge Graphs (PKG). We define PkG as *a shared, multi-layered abstraction of a domain application linked to resources (e.g. services or data) and instructions (e.g. source code or configuration files) in a concrete system*. Specifically, we argue that PKGs have an important role in facing crucial challenges of modern AI applications:

1. PKGs can *unpack* opaque resources (such as source code) into smaller units of analysis, making hidden semantics emerge.
2. Complex LLM-based tasks can be broken down into smaller units and then *repacked* into a more transparent pipeline with explicit semantics.

The rest of the article is structured as follows. We first remind the reader about PKGs, referencing some key research in this area (Section 2). Next, we look into two complementary scenarios involving different hybrid AI systems. In Section 3, we show how a PKG can *unpack* meaning hidden in the source code, considering the case of a typical machine learning training pipeline. Next, we present a scenario where a Large Languege Models (LLM) plays a role in automating a data integration activity such as Knowledge Graph Construction (KGC). We discuss limitations that, in our view, can only be addressed by breaking down KGC in multiples, smaller tasks, and then *repacking* them into a PKG (Section 4). These two scenarios illustrate characteristics that are paradigmatic of modern AI. We summarise the potential of PKG in Section 5. We then discuss limitations and open research challenges in Section 6, before concluding the article (Section 7).

E-mail address: enrico.daga@open.ac.uk.
URL: https://www.enridaga.net.
[1] Research Fellow.

https://doi.org/10.1016/j.websem.2024.100846
Received 17 November 2024; Received in revised form 8 December 2024; Accepted 13 December 2024
Available online 19 December 2024

2. Process knowledge graphs

We consider process knowledge as any formal abstraction of an automated or semi-automated activity. Examples include provenance graphs and workflows. A provenance graph allows the analysis of the lineage of a data object, thus showing, for instance, what data sources have been used to produce a certain digital artefact and who was responsible for it [12,13]. A workflow model represents actions, steps, and agents to allow engineers to reuse software objects and, therefore, repurpose the same implementations in multiple system instances, each one solving a different, real-world scenario [14,15]. Process knowledge representation has a long-standing tradition in computer science research [12,14,16], including in the Ontology Engineering and Semantic Web communities, where such representations are expressed as Knowledge Graph (KG) [17] with the Resource Description Framework (RDF) [18]. RDF KGs have several advantages in relation to interoperability, being based on well-established industry standards. Specifically, they incorporate a uniform naming system (IRI [19]) and a schema-less meta-model (RDF), making them perfect candidates for tasks such as data integration and sharing.

Process Knowledge Graphs (PKG) have been proven essential for:

- describing the provenance of digital objects [13,20]
- supporting analysis and reuse in software engineering (workflows) [21,22]
- support reproducible, open science practices [23,24] and aggregation of experimental metadata [25]
- drive design choices for the realisation of complex or distributed systems (semantic web services) [26]
- support AI explain-ability [27,28]
- support the analysis of complex data pipeline, e.g. when training machine learning models [29]
- supporting software analysis and evolution [30]
- analyse how data processes relate to concerns such as intellectual property or terms and conditions of data licences [31].

Process Knowledge Graphs (PKG) can help describe complex, neuro-symbolic, and hybrid human-machine applications to facilitate the understanding of such systems from the point of view of how data is consumed, processed, and repurposed.

Beyond the numerous applications, we argue that the power of PKGs stands in providing a solution to two fundamental tasks, which are crucial ones in modern AI-based applications:

(i) the understanding of *opaque* automation, which we refer to as *unpacking*, and
(ii) the control of AI behaviour via decomposition and reassembly, which we refer to as *repacking*.

In what follows, we report on research conducted to *unpack* data science pipelines with PKGs. Next, we discuss a frontier scenario regarding the use of PKG to better control an LLM, as well as opportunities and challenges.

3. Unpacking AI applications: an example with data science pipelines

Artificial intelligence systems are not constructed from just a single dataset or trained model. Instead, they are developed through intricate data science workflows that incorporate multiple datasets, models, preparation scripts, and algorithms. Due to this complexity, comprehending these AI systems necessitates providing explanations at higher levels of abstraction.

In [29], we argue how a PKG based on the Data Journeys Ontology (DJO), can provide such abstraction. There, we consider the case of code designed to train machine learning models, typically, implemented in Python *notebooks*, exemplified by the code snippet

```
1 import pandas as pd
2 # Load data
3 melbourne_file_path = '../input/melbourne-housing-
      snapshot/melb_data.csv'
4 melbourne_data = pd.read_csv(melbourne_file_path)
5 # Filter rows with missing values
6 melbourne_data = melbourne_data.dropna(axis=0)
```

Fig. 1. Example of python code.

in Fig. 1, taken from a well-known repository of machine learning pipelines.[2]

Line 1 and 3 mention references to *resources*: a software library (`pandas`) and a CSV file. Then, each line performs some activity. DJO serves as a multi-layered semantic representation of data processing activities associated with data science code and assets. The approach observes the activity using the following layered methodology:

Resources These are the digital artefacts utilised in the data journey, such as source code files, software libraries, services, or data sources.

Source Code This element comprises instructions that are both human-readable and machine-executable, like a Python script.

Machine Representation This expression refers to any form of machine-interpreted representation of the instructions, such as an Abstract Syntax Tree (AST) or a query execution plan.

Datanode Graph Defined in [31], this graph represents the relationships between data elements, including variables, imported libraries, and input/output resources. This abstraction highlights the structure of data flows, concentrating on data-to-data dependencies while setting aside issues related to control flow.

Activity Graph This graph depicts high-level activities and is inspired by the idea of Workflow Motifs [21].

In the above description, DJO would contribute the last two layers, thus providing a high-level abstraction.

Considering the example above, a datanode graph would *unpack* the intended data flow, providing a first, granular abstraction linked to the variables and operations in the source code (see Fig. 2). In this representation layer, variables are nodes, while arcs represent operations such as assignments, function calls, and methods. It is important to note that such representation is not a complete translation of the source code. Control elements, such as iterations and conditions, are removed from the picture. What remains is a *data flow* that can be traversed, explored, and used to explore semantic relationships between input and outputs of the data science pipeline. However, many operations could be grouped together into high-level activities, such as in the activity layer as represented in Fig. 3. DJO focuses on data flows and high-level activities, with the objective of helping explain how machine learning models are produced. Further details on this example, and on DJO in general, can be obtained from [29]. Here, it is worth considering how PKG allow the incorporation of multiple, potentially open-ended concerns within a unified representation strongly linked to concrete, low-level elements of the system.

4. Towards *repacking* AI applications

In this section, we discuss how PKGs offer a principled way of breaking down generative AI tasks into smaller units, thus improving transparency through explicit semantics. We consider a typical application of generative AI and analyse it in the light of opportunities provided by PKGs.

[2] Kaggle https://www.kaggle.com/code/dansbecker/random-forests (Accessed on 16/11/2024).

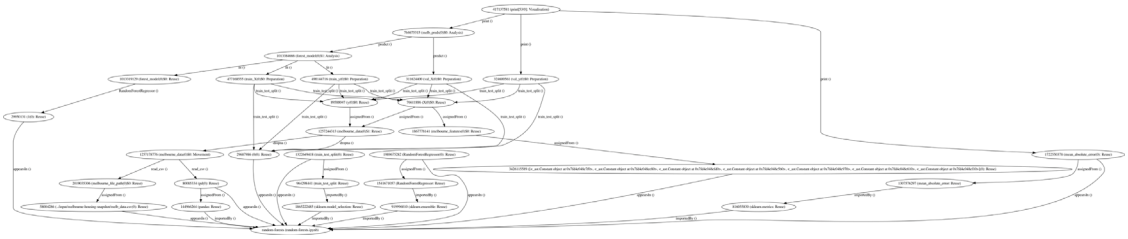

Fig. 2. Datanode graph of the random-forests data journey, used in the example. Such complex representation constitutes already an abstraction over the source code, by removing control structures such as iterations, conditions, and function calls, and leaving the data (as nodes) and the operations (as arcs).

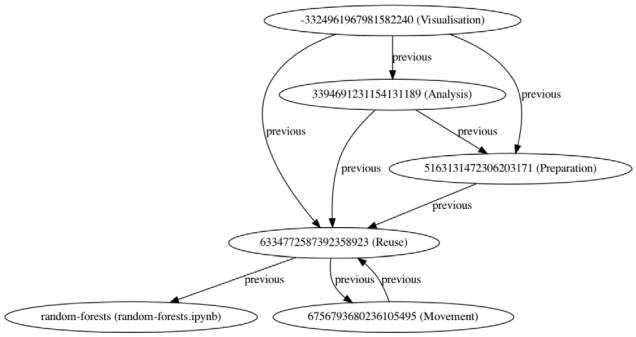

Fig. 3. Example of high level activities derivable from the python code snippet. This representation constitutes another layer of the PKG, together with the datanode layer, illustrated in Fig. 2. This allows to traverse the graph from an activity, to all the data nodes involved and, finally, to the elements of the source code responsible for it.

It is established knowledge that generative AI applications based on a single, one-shot prompt have a high risk of producing wrong content, either because of erroneous interpretation of the input or because of hallucinating [32]. Chain of thought [33] has initiated a line of approaches where prompt engineering is articulated in multiple steps, sometimes reproducing pseudo-logical reasoning steps such as in [34,35], with the attempt of increasing the chance of accuracy and reducing the risk of hallucination. Such approaches to prompt engineering emerged at the same time with a variety of software tools that allow developers to redesign a complex task in smaller units and interact with the LLM in multiple stages, with the attempt of increasing the chance of accuracy and reducing the risk of hallucination [5,6]. Here, we argue that PKG can provide the backbone for a principled decomposition of opaque, black-box tasks into smaller units, and help the coordination and reuse of generative AI-powered applications.

A paradigmatic case is the one of code generation, a task in which current generative AI performs particularly well with very popular programming languages (Codex,[3] Copilot,[4] and others). A special case is automating Knowledge Graph Construction (KGC). In summary, the task can be expressed as follows: *given an ontology and a data source in a format which is not RDF, generate a knowledge graph, expressed in Turtle*. Intuitively, using an LLM *as-is* for this task would be problematic, mainly because most real-world scenarios have data of sizes that go well beyond what is supported by input prompts. A slightly different (but more realistic) way of characterising the task for an LLM would be: *given an ontology and a (sample) data source in a format which is not RDF, generate code that, when executed, will generate the desired knowledge graph* [36]. Research is emerging to tackle this problem, for example, by studying the generation of RML mappings [37]. Others are investigating how well LLMs can interpret and generate Turtle [38], considered the most human-readable RDF syntax.

[3] OpenAI Codex: https://openai.com/index/openai-codex/.

[4] Github Copilot: https://github.com/features/copilot.

Preliminary experiments performed in the context of a doctoral research [36] are showing how, while LLMs are very good in simple mapping operations – e.g. associating a schema term such as *dateOfBirth* to the Schema.org property *birthDate* – they fall short in several other aspects, such as guessing the right datatype or handling missing information. In addition, complex remodelling tasks, such as breaking down a data unit into multiple entities, require a different approach to prompt engineering. For example, when a CSV row includes a person's details, an address, a car registration number, and so on, multiple interlinked entities need to be generated. Finally, many decisions in KGC are actually *design choices*, such as the pattern to use when building IRIs.

Looking at the task above as a human activity can help us understand why it is so challenging for LLMs to perform it. In that scenario, developers typically observe the terms in the target ontology and map them to the ones in the source data. Then, they design a transformation procedure (with a programming language or tool they prefer). By executing it, they can evaluate whether the output is as intended. If not, they can modify the transformation procedure until they obtain the desired output [39]. To achieve that goal, developers perform many tasks. They choose how to map the schema terms to the ontology. They decide which data point becomes an entity or a simple literal. They judge what data types to associate with the literal values. They establish design solutions for IRIs and apply structural patterns (e.g. for N-Ary relations).

We can consider a well-known source of design patterns for linked data publishing (the *Linked Data Patterns Book*, [40]) as an inspiration for how we could break down the KGC task into smaller units, each one of them associated to a specific prompt and organised in a rich, detailed workflow that could be implemented with state of the art LLM application development tool. We are not doing this right now, but only observe that the cited book lists more than 50 patterns that users may consider. KGC requires reorganising the operation from a single prompt to a collection of atomic ones and strategies to combine them in meaningful processes. PKGs have a role in handling complex scenarios like this one.

5. Opportunities

In summary, PKGs provide a number of opportunities for *unpacking* and *repacking* complex AI applications:

- **PKGs provide the building blocks for describing complex automations at different levels of abstractions** [25,29,41,42]. PKGs offer a range of ontologies to express activities, resources, and their relationship, which can be extended to fit concrete applications.
- **PKGs can be multi-layered**, while keeping robust referencing to concrete artefacts (data, software), they augment them with metadata in a separate representation, therefore, allowing the co-existence of multiple, possibly conflicting perspectives on the same systems. Such descriptions will be increasingly relevant to answer current socio-technical problems in deploying AI applications in the real world, which raise concerns related to trustworthiness [43], accountability [44], bias [45], and ethics [46].

- **PKGs can express process designs** (the intentions behind a certain process) **but also process executions** (what a system has actually done). In this way, they can be used to analyse AI applications behaviours, supporting key operations such as auditing [42,47] and forensics [48].
- By relying on established semantic web principles, **PKGs can provide a reference architecture for collecting, sharing, and reuse** of LLM-powered tasks. This would allow the design LLM-powered operations independently from concrete software implementations as well as going beyond existing registries of prompts as templates.[5]

6. Limitations and challenges

So far, we have illustrated the benefit of PKGs for AI system design. However, current PKGs come with limitations:

- **Domain modelling** effort is required to tailor the PKG to the specific application. In our examples, we used workflow motifs for data science pipelines [21] and linked data patterns for KGC [40]. However, every application is different.
- The effort required to **populate the PKG** cannot be underestimated. In our examples, specific algorithms transformed Python code into a datanode graph [29]. Applying the same method to other languages would require dedicated software.
- Applications are implemented with various software and data artefacts, including subsystems such as databases, services and APIs, and those exposing ML models such as LLMs. The **automatic or semiautomatic extraction of complex workflows** spanning different artefacts is a challenging problem.
- KGs are a special type of databases. Nevertheless, they are representations of something that can **change over time**. Therefore, keeping the PKG up to date and adapting it to changes occurring in the reference application is an important issue. Without the guarantee that knowledge is up to date, the resulting solution will not be trustworthy.

In the light of these limitations, we can sketch a list of open challenges ahead for the KG research community:

1. Eliciting process knowledge from existing artefacts such as software code is difficult. Relevant data may be extracted from design documents, software artefacts, configuration files, execution trace logs, and so on. Despite efforts for augmenting artefacts with metadata [24], **streamlining the extraction and integration of process knowledge is still an open challenge**.
2. Research on social aspects of AI (e.g. bias, trust, accountability), which address governance of AI applications, are in constant evolution. While PKGs can provide the technical foundation for knowledge engineering in this area (e.g. [49]), an integrated approach would necessarily require interdisciplinary expertise and involve multiple communities (e.g. from business, engineering, and the law). **The community should reflect on how different perspectives could be integrated into a unified representation**, the DJO multi-layered approach could provide a reference pattern [29].
3. Despite the numerous ontologies dedicated to process knowledge, it is an open question to what extent it would be possible to **generalise over the variety of AI applications** and describe them in their constitutional design properties so that their trustworthiness, quality, and reuse can be maximised.

[5] E.g. what is provided by platforms such as Promptlayer https://docs.promptlayer.com/introduction.

7. Conclusions

Process knowledge is fundamental for *studying* and *designing* AI applications under a consistent methodological framework, paving the way to a more agile and principled integration of symbolic and sub-symbolic methods in AI. Grounded on well-established knowledge representation good practices, PKGs have the potential to offer a solution for making sense of complex AI systems and for aiding the design of complex AI automations. However, knowledge modelling, population, and evolution, still require to be addressed. Finally, within a fast-changing socio-technical landscape, a multi-disciplinary approach is fundamental for achieving a robust and trustworthy technical solution. The KG community seems to be the right one to take up the challenge.

Declaration of competing interest

Author declare that there are no conflicts of interest related to this submission.

Data availability

No data was used for the research described in the article.

References

[1] S.S. Sengar, A.B. Hasan, S. Kumar, F. Carroll, Generative artificial intelligence: A systematic review and applications, Multimedia Tools Appl. (2024) 1–40.
[2] I. Ghebrehiwet, N. Zaki, R. Damseh, M.S. Mohamad, Revolutionizing personalized medicine with generative AI: a systematic review, Artif. Intell. Rev. 57 (5) (2024) 1–41.
[3] L. Stappen, J. Dillmann, S. Striegel, H.-J. Vögel, N. Flores-Herr, B.W. Schuller, Integrating generative artificial intelligence in intelligent vehicle systems, in: 2023 IEEE 26th International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2023, pp. 5790–5797.
[4] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, D.C. Schmidt, A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023, arxiv preprint arXiv:2302.11382.
[5] O. Topsakal, T.C. Akinci, Creating large language model applications utilizing langchain: A primer on developing llm apps fast, in: International Conference on Applied Engineering and Natural Sciences, vol. 1, (no. 1) 2023, pp. 1050–1056.
[6] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S.K.S. Yau, Z. Lin, L. Zhou, et al., Metagpt: Meta programming for multi-agent collaborative framework, 2023, arxiv preprint arXiv:2308.00352.
[7] C. d'Amato, L. Mahon, P. Monnin, G. Stamou, Machine learning and knowledge graphs: Existing gaps and future research challenges, Trans. Graph Data Knowl. 1 (1) (2023) 1–35.
[8] A. Breit, L. Waltersdorfer, F.J. Ekaputra, M. Sabou, A. Ekelhart, A. Iana, H. Paulheim, J. Portisch, A. Revenko, A.t. Teije, et al., Combining machine learning and semantic web: A systematic mapping study, ACM Comput. Surv. 55 (14s) (2023) 1–41.
[9] F. Lecue, On the role of knowledge graphs in explainable AI, Semant. Web 11 (1) (2020) 41–51.
[10] B. Zhang, V.A. Carriero, K. Schreiberhuber, S. Tsaneva, L.S. González, J. Kim, J. de Berardinis, OntoChat: a framework for conversational ontology engineering using language models, in: Proceedings of the Extended Semantic Web Conference, ESWC, 2024, arxiv preprint arXiv:2403.05921.
[11] R. Alharbi, V. Tamma, F. Grasso, T.R. Payne, Investigating open source LLMs to retrofit competency questions in ontology engineering, in: Proceedings of the AAAI Symposium Series, vol. 4, (no. 1) 2024, pp. 188–198.
[12] P. Buneman, S. Khanna, W.-C. Tan, Data provenance: Some basic issues, in: FST TCS 2000: Foundations of Software Technology and Theoretical Computer Science: 20th Conference New Delhi, India, December 13–15, 2000 Proceedings 20, Springer, 2000, pp. 87–93.
[13] T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, J. Zhao, Prov-o: The prov ontology, W3C Recommend. 30 (2013).
[14] D. Georgakopoulos, M. Hornick, A. Sheth, An overview of workflow management: From process modeling to workflow automation infrastructure, Distrib. Parallel Databases 3 (1995) 119–153.
[15] W.M. van Der Aalst, A.H. Ter Hofstede, B. Kiepuszewski, A.P. Barros, Workflow patterns, Distrib. Parallel Databases 14 (2003) 5–51.
[16] E.A. Stohr, J.L. Zhao, Workflow automation: Overview and research issues, Inform. Syst, Front. 3 (2001) 281–296.

[17] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G.D. Melo, C. Gutierrez, S. Kirrane, J.E.L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, ACM Comput. Surv. (Csur) 54 (4) (2021) 1–37.
[18] D. Wood, M. Lanthaler, R. Cyganiak, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation, W3C, 2014, https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.
[19] M. Dürst, M. Suignard, Internationalized Resource Identifiers (IRIs), Technical Report, 2005.
[20] S. Ram, J. Liu, et al., A new perspective on semantics of data provenance, SWPM 526 (2009).
[21] D. Garijo, P. Alper, K. Belhajjame, O. Corcho, Y. Gil, C. Goble, Common motifs in scientific workflows: An empirical analysis, Future Gener. Comput. Syst. 36 (2014) 338–351.
[22] R. Bergmann, Y. Gil, Similarity assessment and efficient retrieval of semantic workflows, Inf. Syst. 40 (2014) 115–127.
[23] Ó. Corcho, D. Garijo Verdejo, K. Belhajjame, J. Zhao, P. Missier, D. Newman, R. Palma, S. Bechhofer, E. García Cuesta, J.M. Gomez-Perez, et al., Workflow-Centric Research Objects: First Class Citizens in Scholarly Discourse, Informatica, 2012.
[24] M.R. Crusoe, S. Abeln, A. Iosup, P. Amstutz, J. Chilton, N. Tijanić, H. Ménager, S. Soiland-Reyes, B. Gavrilović, C. Goble, et al., Methods included: standardizing computational reuse and portability with the common workflow language, Commun. ACM 65 (6) (2022) 54–63.
[25] I. Dasoulas, D. Yang, A. Dimou, MLSea: A semantic layer for discoverable machine learning, in: European Semantic Web Conference, Springer, 2024, pp. 178–198.
[26] L. Cabral, J. Domingue, E. Motta, T. Payne, F. Hakimpour, Approaches to semantic web services: an overview and comparisons, in: The Semantic Web: Research and Applications: First European Semantic Web Symposium, ESWS 2004 Heraklion, Crete, Greece, May 10-12, 2004. Proceedings 1, Springer, 2004, pp. 225–239.
[27] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, W. Samek, Explainable AI methods-a brief overview, in: International Workshop on Extending Explainable AI beyond Deep Models and Classifiers, Springer, 2022, pp. 13–38.
[28] I. Tiddi, S. Schlobach, Knowledge graphs as tools for explainable machine learning: A survey, Artificial Intelligence 302 (2022) 103627.
[29] E. Daga, P. Groth, Data journeys: Explaining AI workflows through abstraction, Seman. Web 15 (4) (2024) 1057–1083.
[30] J. Tappolet, C. Kiefer, A. Bernstein, Semantic web enabled software analysis, in: Web Semantics: Science, Services and Agents on the World Wide Web, vol. 8, (no. 2–3) Elsevier, 2010, pp. 225–240.
[31] E. Daga, M. d'Aquin, A. Gangemi, E. Motta, Propagation of policies in rich data flows, in: Proceedings of the 8th International Conference on Knowledge Capture, 2015, pp. 1–8.
[32] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al., A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, ACM Trans. Inf. Syst. (2023).
[33] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q.V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, in: Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 24824–24837.
[34] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, et al., Self-refine: Iterative refinement with self-feedback, Adv. Neural Inf. Process. Syst. 36 (2024).
[35] Z. Ling, Y. Fang, X. Li, Z. Huang, M. Lee, R. Memisevic, H. Su, Deductive verification of chain-of-thought reasoning, Adv. Neural Inf. Process. Syst. 36 (2024).
[36] M. Ratta, AI supported knowledge graph design & generation, in: PhD Symposium of the Extended Semantic Web Conference, ESWC, in: 21st International Conference, ESWC 2024, Hersonissos, Crete, Greece, May 26–30, 2024, Proceedings, Part II, 2024.
[37] M. Hofer, J. Frey, E. Rahm, Towards self-configuring knowledge graph construction pipelines using llms-a case study with rml, in: Fifth International Workshop on Knowledge Graph Construction@ ESWC2024, 2024.
[38] J. Frey, L.-P. Meyer, N. Arndt, F. Brei, K. Bulert, Benchmarking the abilities of large language models for rdf knowledge graph creation and comprehension: How well do llms speak turtle? in: Proceedings of the DL4KG Workshop, co-located with the International Semantic Web Conference (ISWC), 2023, arxiv preprint arXiv:2309.17122.
[39] L. Asprino, E. Daga, A. Gangemi, P. Mulholland, Knowledge graph construction with a façade: a unified method to access heterogeneous data sources on the web, ACM Trans. Internet Technol. 23 (1) (2023) 1–31.
[40] L. Dodds, I. Davis, Linked data patterns, 2011, Online: http://patterns.dataincubator.org/book.
[41] A. Harth, T. Käfer, A. Rula, J.-P. Calbimonte, E. Kamburjan, M. Giese, Towards Representing Processes and Reasoning with Process Descriptions on the Web, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.
[42] F.J. Ekaputra, A. Ekelhart, R. Mayer, T. Miksa, T. Šarčević, S. Tsepelakis, L. Waltersdorfer, Semantic-enabled architecture for auditable privacy-preserving data analysis, Semant. Web 15 (3) (2024) 675–708.
[43] B. Li, P. Qi, B. Liu, S. Di, J. Liu, J. Pei, J. Yi, B. Zhou, Trustworthy AI: From principles to practices, ACM Comput. Surv. 55 (9) (2023) 1–46.
[44] C. Novelli, M. Taddeo, L. Floridi, Accountability in artificial intelligence: what it is and how it works, Ai Soc. 39 (4) (2024) 1871–1882.
[45] R. Schwartz, R. Schwartz, A. Vassilev, K. Greene, L. Perine, A. Burt, P. Hall, Towards a Standard for Identifying and Managing Bias in Artificial Intelligence, vol. 3, US Department of Commerce, National Institute of Standards and Technology, 2022.
[46] T. Hagendorff, The ethics of AI ethics: An evaluation of guidelines, Minds Mach. 30 (1) (2020) 99–120.
[47] J. Mökander, Auditing of AI: Legal, ethical and technical approaches, Digit. Soc. 2 (3) (2023) 49.
[48] J. Schneider, F. Breitinger, AI forensics: Did the artificial intelligence system do it? why, 2020, arxiv preprint arXiv:2005.13635.
[49] M. Sabou, M. Llugiqi, F.J. Ekaputra, L. Waltersdorfer, S. Tsaneva, Knowledge engineering in the age of neurosymbolic systems, Neurosymbolic AI J. (submitted for publication) (2024).

Enrico Daga has carried out R&D on Web Semantics first at the Italian National Research Council (CNR) and then at the Knowledge Media Institute of The Open University in the UK, where is currently Senior Research Fellow. His research is exploring novel methods for Knowledge Graph generation and infrastructure, with particular focus on data management, policies, and process knowledge. He applies his expertise to data-intensive scenarios in the smart cities, healthcare, and cultural heritage domains.