Towards Efficient Field Service Engineering for Powertrains via LLM-generated Knowledge Graphs

Prerna Juhlin[1,*], Rana Hussein[1] and Nicolai Schoch[1]

[1]ABB AG Corporate Research Center Germany, Kallstadter Strasse 1, D-68309 Mannheim, Germany

Abstract

A knowledge integration framework is presented for efficient field service engineering in the context of powertrains. The framework enables semantic integration of otherwise siloed data from different operations and maintenance system databases. Starting with an expert-curated and database schema-aligned ontology, and using LLMs, a knowledge graph is created from unstructured data sources. A comparison of two leading LLM-based approaches is provided for unstructured data integration in knowledge graphs, namely LangChain LLM Graph Transformer and Microsoft GraphRAG. We also suggest customizations for fine-tuning of the generation process. Besides efficient powertrain fault resolutions, potential applications include Root Cause Analysis (RCA), Failure Mode and Effects Analysis (FMEA), as well as prescriptive maintenance.

Keywords

powertrain, field service engineering, RCA, FMEA, prescriptive maintenance, LLM Graph Transformer, GraphRAG

1. Introduction

A powertrain is a complex asset consisting of a motor, a drive and a load, with a typical lifecycle covering design, engineering, operations and service involving multiple heterogeneous tools. State-of-the-art digital services such as ABB Ability™ Digital Powertrain integrate data from sensors and drives together with cloud-based analytics for monitoring and predictive maintenance services [1]. Beyond sensor and equipment data collecting real-time operational values, Field Service Engineers (FSEs) typically also use information from service manuals, installed base database, Customer Relationship Management (CRM) system and Enterprise Asset Management (EAM) system to perform field service and maintenance activities. Providing efficient support to customers is of utmost priority for FSEs, however the normally siloed nature of the different information sources means that finding relevant information for service tasks can be highly time-consuming and is recognized as a top concern [2]. Furthermore, the composite asset structure of a powertrain include a drive, a motor, and a load, with information that that may be asset-type specific, adds another layer of complexity not easily managed through manual means.

A semantic layer in the form of a knowledge graph connecting different data points with meaningful relationships can provide integrated information in context accessible to FSEs for troubleshooting, failure analyses and performing corrective actions. However, scaling knowledge graph generation to hundreds of thousands of data points is not manually feasible. Whereas Extract-Transform-Load (ETL) pipelines have been successfully used for structured datasets [3], use of the novel technology of Large Language Models (LLMs) have been recently proposed when dealing with large amounts of unstructured data which LLMs are inherently more capable of working with.

The importance of knowledge management for field service engineering has been previously recognized and an ontology-based approach has been presented in [4]. To the best of the authors' knowledge, this paper presents a first evaluation of LLM usage for knowledge graph generation in this context. The remainder of the paper is organized as follows: Section 2 presents an overview of the powertrain knowledge integration framework with a novel semantic integration layer together with an outline

SGKi 2025: Scaling Knowledge Graphs for Industry Workshop, co-located with SEMANTiCS'25: International Conference on Semantic Systems, September 3–5, 2025, Vienna, Austria

*Corresponding author.

prerna.juhlin@de.abb.com (P. Juhlin); rana.hussein@de.abb.com (R. Hussein); nicolai.schoch@de.abb.com (N. Schoch)

0000-0003-4815-129X (P. Juhlin); 0009-0000-2572-859X (R. Hussein); 0000-0002-7806-6435 (N. Schoch)

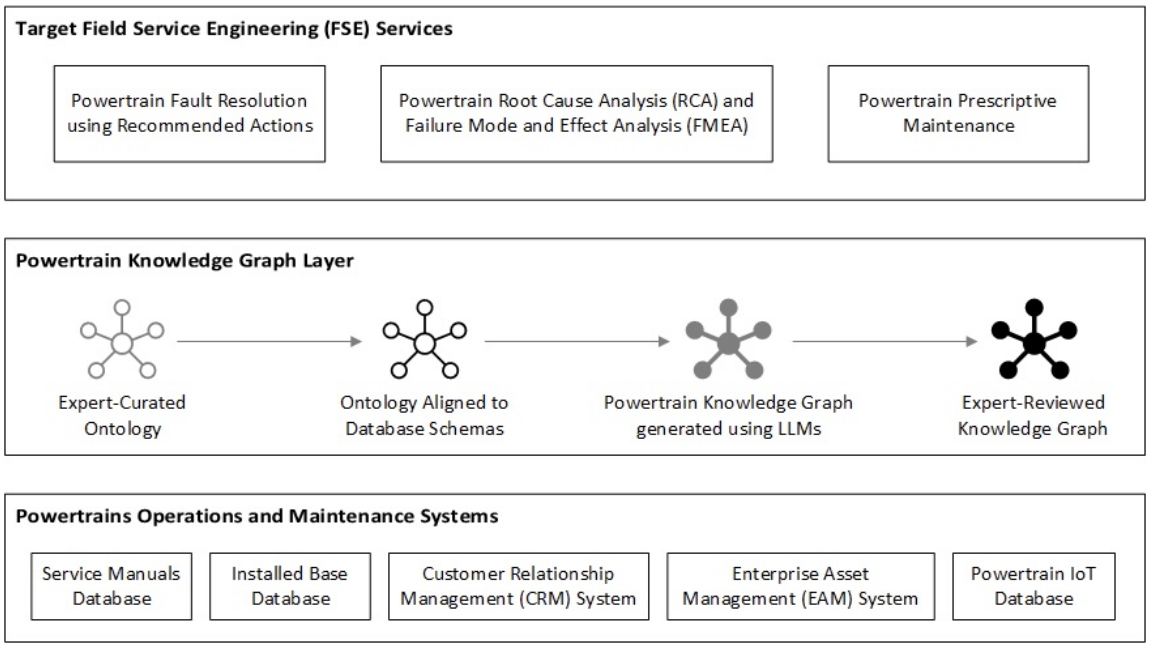

Figure 1: Powertrain Knowledge Integration Framework with Knowledge Graph and Potential FSE Services

for different FSE applications; Section 3 presents a comparison of leading LLM-based approaches for unstructured data together with a comparative evaluation, findings and related customizations; Section 4 concludes the paper with remarks and a future outlook.

2. Powertrain Knowledge Integration Framework

The overall powertrain knowledge integration architecture is organized in layers as shown in Figure 1. Different information such as service instructions, installed base data, service history, customer requests, maintenance history and operational data are available via the various specialized databases from multiple vendors in 'Powertrain Operations and Maintenance Systems'. Due to the siloed information sources and heterogeneous data models, relations across information entities can easily be missed. Field service engineers looking for information about a particular asset on site must refer to various databases and manually find relevant interconnections, which tends to be a time-consuming and error-prone process. As a result, services such as fault resolution and root cause analyses requiring integrated information from different operations and maintenance systems are difficult to perform.

The 'Powertrain Knowledge Graph Layer' consists of an ontology with TBox statements and a knowledge graph with instances corresponding to ABox statements [5], based on the underlying data sources for powertrain. The ontology is created based on expert curation of classes and relationships, which is subsequently aligned to type- and attribute-level information from different database schemas. In particular, relationships across information entities not captured in any single database are possible to represented in the ontology based on expert knowledge. For example, service cases in a CRM system about a certain drive product family are linked via the drive fault code to recommended actions in the corresponding drive service manual. The ontology is created using the W3C Resource Description Framework (RDF) family of standards [6], which allows semantic representations as well as the possibility to process using LLMs, e.g., in Turtle syntax [7]. The ontology serves as a basis for constructing the 'Powertrain Knowledge Graph' using LLMs for unstructured data sources, as described in Section 3. Due to the potential for errors in LLM outputs, which can vary based on selected approach as explained in Section 3, the generated knowledge graph is reviewed by experts for quality assurance in a final step.

Troubleshooting, resolution and maintenance are among the primary tasks of FSEs, see e.g., [8, 9]. Related use cases requiring integrated information are enabled based on the constructed knowledge

graph in combination with querying, e.g., using the RDF query language SPARQL [10]:

1. `Fault Resolution` FSEs dealing with faulty equipment require assessing the current situation, considering operational history and maintenance records, and performing relevant standard checks and procedures. Based on the connected information stored in the knowledge graph, the process can be guided through resolution step based on recommended actions from the service manuals. In case of a novel resolution means not hereto recorded in the manuals, the fix can be stored in the knowledge graph as future reference for future cases.
2. `Root Cause Analysis and Failure Mode and Effects Analysis (FMEA)` Successful failure resolution typically also involves performing the Root Cause Analysis (RCA) for preventive maintenance [11]. The knowledge graph provides operational data, case history, maintenance records and service instructions from manuals with relevant connections, enabling easier access to information with their context for efficient investigations. The knowledge graph can also enable design-phase FMEA used to anticipate failure in an early phase based on the integration of history across operations and maintenance together with actual service resolutions and service instructions, recorded for different powertrain instances. FSE feedback can be recorded in the knowledge graph for a constantly up-to-date integrated knowledge base.
3. `Powertrain Prescriptive Maintenance` Beyond preventive maintenance, prescriptive maintenance services could also be provided based on mapped future scenario possibilities with related instructions or resolutions stored in the knowledge graph.

3. LLM-based Approaches for Knowledge Graph Generation

Field service engineers typically rely on service manuals to perform maintenance tasks. These manuals are in the form of unstructured data and are usually complex containing domain specific concepts and terminology. Capturing the meanings in such unstructured text and understanding the connections between different entities is not a trivial task. Large Language Models (LLMs) have recently shown significant capabilities in several natural language processing research areas. Among these areas is converting unstructured data into entities and relationships to construct knowledge graphs for large amounts of unstructured text, e.g., as investigated in [12]. The LLM-based approaches are particularly valuable when it comes to scaling the work of structuring large amounts of unstructured texts [12]. A well-known challenge is that LLM models are prone to hallucinations [13, 14], i.e. LLMs may fabricate responses that are factually incorrect. As a result, the knowledge graph can contain an inaccurate representation of service instructions which hinders the quality of maintenance activities.

In this section, we explore different LLM techniques and frameworks to convert unstructured text into knowledge graphs. We highlight our findings and present customizations to overcome some challenges.

3.1. LLM Techniques

We investigate three techniques to explore LLM capabilities in converting large amounts of unstructured text to entities and relationships.

3.1.1. Schema-less technique

A schema-less technique is an unstructured approach that allows LLMs to build knowledge graphs without relying on any pre-defined graph structure. Here, the input to the LLM is only the unstructured text. The LLM subsequently generates a schema representing the underlying entities and relationships and populates instances accordingly.

3.1.2. Schema-based technique

In a schema-based technique, a structured representation of the TBox statements in the form of an ontology is provided to the LLM to act as a guide for capturing entities and relationships. The objective

of the LLM is to find triples in the text conforming with the given ontology schema. Here, the expert-curated ontology aligned to database schemas in a second step, as shown in Figure 1, is provided. Example nodes include fault codes, which are identifiers representing specific faults generated by a drive or a motor as well as likely causes and instruction explaining how to resolve such faults. Examples of relationships include linking a drive product family to a fault code, by specifying that a drive generates a fault code and a fault code occurs due to a likely cause.

3.1.3. Prompt design

Providing additional prompt instructions is a widely used approach to provide clear guidance to the LLM [15, 16]. This can further enhance the generation of more accurate responses and reduce hallucinations. We investigate the benefit of providing such prompts by designing a set of detailed instructions that contain contextual information and demonstrations of domain specific input and output examples. An example of such instructions is including a fault code number representation in the entity text. We provide examples to guide the LLM on how to handle unseen fault codes encountered in unstructured text. Handling such domain specific details is not obvious without providing additional instructions. Another example is instructing the LLM to include the full text of a sentence in the entity text as opposed to summarizing or cutting the sentence. This is crucial in our case, as the text often contains detailed instructions from a manual on how to address a particular drive fault.

3.2. LLM Frameworks

We investigate two widely used LLM frameworks to generate knowledge graphs: LangChain LLM Graph Transformer and Microsoft GraphRAG. To drive a fair comparison, we use GPT-4 model [17] for both frameworks.

3.2.1. LLM Graph Transformer

LLM Graph Transformer [18] is a framework developed by Langchain [19] and Neo4j [20]. It extracts structured information from text using LLMs. Text is split into several chunks according to a given window size. LLM models are then used to extract entities and relationships from text chunks. The extracted information is stored in Neo4j graph database to facilitate downstream RAG applications. LLM Graph Transformer provides several customizations to tailor the LLM depending on a given use case. First, it provides the ability to define a graph schema in a precise way, which provides guidance and structure to the LLM. The graph schema is a set of entity types and relationships in the form of triples: source node, relationship type, and target node. Second, the framework has the flexibility to use a variety of LLM models. Third, a prompt with additional instructions can be passed to the LLM for further customization.

3.2.2. Microsoft GraphRAG

Microsoft's GraphRAG [21] extracts structured entities and relationships from natural language text using LLMs. A graph index is built and an entity knowledge graph is derived. GraphRAG also leverages global information in the graph by using community detection to cluster closely related communities together. This is then used for summarization and RAG retrieval applications. This framework supports providing an ontology to guide the structuring of the knowledge graph. Several prompts are provided for extracting the graph, clustering communities, and performing local and global searches on the knowledge graph.

3.3. Findings

Below we present our findings for the three LLM techniques used to construct the knowledge graph. We also compare the two LLM frameworks using several aspects.

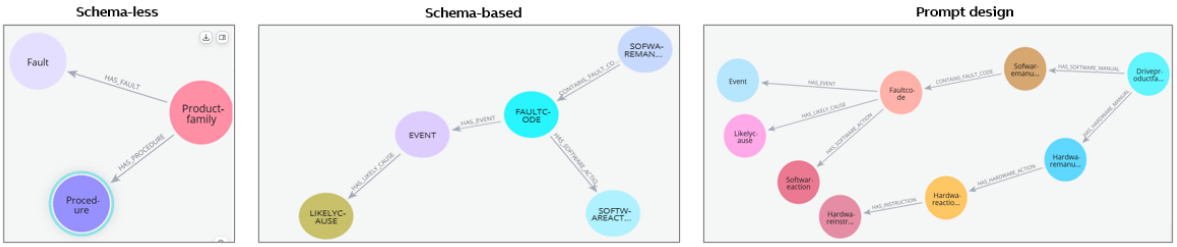

Figure 2: Schemas generated from using different LLM techniques

3.3.1. Evaluating LLM techniques

We evaluate the knowledge graph generated from unstructured text in manuals. The manuals contain a description of faults encountered by drives, their likely causes, and resolution steps. Figure 2 shows the schemas generated from LLM models when using each of the three LLM techniques. First, for the schema-less technique, although the LLM has more freedom and flexibility to construct the knowledge graph, we observe that it fails in capturing the underlying information in the unstructured text. This leads to misrepresenting entities and relationships in the knowledge graph as well as ignoring relevant concepts in the text, such as likely causes of drive faults. Second, in the schema-based technique, we observe a more informative structure in the knowledge graph. The entities and relationships accurately represent the domain information present in the unstructured text. However, we still notice the LLM missing essential semantic connections present in the text. Finally, when providing detailed prompt instructions in addition to the ontology, we observe an improvement in the quality of the generated schema and instances. The LLM is able to accurately capture the required domain knowledge in the text and form meaningful semantic connections. For example, fault code entity representation accurately follows the provided prompt examples. This is crucial for service engineers to be able to accurately recognize faults generated by drives. Also, when providing instructions to the LLM, full instructions from the manuals are included in the instance graphs. This is important for effective diagnoses and repair of faults.

3.3.2. Evaluating LLM frameworks

We compare both frameworks against several aspects in Table 1. The first aspect is stable runs. Both frameworks use LLMs, as a consequence the graph generation process is non-deterministic which may lead to variant output among different runs. Producing different output is problematic in real-world scenarios where a consistent data generation workflow is required. To assess the magnitude of the variation for both frameworks, we perform 10 runs using the same input and prompts. We examine the consistency of the output in terms of the following: the total number of generated entities/relationships and the number of generated entities per each type in the ontology schema. When using GraphRAG, we notice consistent output across multiple runs. However, when using LLM Graph Transformer, a substantial variation of results is observed. The second aspect of the comparison is related to the schema-based technique. LLM Graph Transfomer offers more control over specifying the ontology structure. A detailed specification of relationship types and properties can be provided to guide the LLM. Next, we notice both frameworks fail to semantically link and understand domain specific terminology. Also, both frameworks are prone to hallucinations. LLM Graph Transformer exhibits a hallucination rate of ∼15%, while for GraphRAG, the hallucination rate is ∼12%. In Section 3.4 we further discuss the concept of hallucinations. Hence, summarizing, our preference based on scalability-relevant criteria such as ‘Stable runs’ and ‘Triple hallucination avoidance’ is to use GraphRAG as it performs better compared to the LLM Graph Transformer approach. Additionally, the ‘Provide summary descriptions’ and ‘Generate communities’ features can aid in the future for processing of the constructed knowledge graph.

Table 1
Comparison of employed LLM Techniques

Feature	LLM Graph Transformer	GraphRAG
Stable runs	×	√
Full control over provided schema	√	×
Semantic equivalence of domain-specific terminology	×	×
Triple hallucination avoidance	×	×
Provide summary descriptions	×	√
Generate communities	×	√

3.4. Customizations

3.4.1. Hallucinations

LLM approaches are prone to hallucinations. We observe several examples of such behavior. First, hallucinations exist in the form of new relationship types that do not exist in the ontology schema. This leads to incorrect semantic linking between entities. Second, hallucinations exist in the form of triple hallucinations. Although the type of relationship as well as the types of generated entities exist in the ontology schema, the LLM fabricates and mixes false associations between entity types that are not consistent with the domain knowledge. Next, we observe cases where the triple types and associations follow the ontology schema, however, the generated instances are not present in the input data. Finally, in some cases, entities are generated with no assigned types. In this case, the LLM fails to understand the meaning of such entities. These examples are flagged in the graph for further domain expert review.

3.4.2. Knowledge Graph verification and alignment to the ontology

Different from hallucinations, LLMs may generate triples that are slightly deviated from the ontology structure. For example, a relationship can be reversed between two entities. We adjust such inconsistencies to ensure that the instances are aligned with the defined ontology concepts. The verification step involves the experts towards the final 'Expert-Reviewed Knowledge Graph' of Figure 1.

3.4.3. Mapping layer

We provide the capability to store the output of the LLM, in the form of entities and relationships, in different RDF-native and labeled property graph (LPG) databases. To achieve this, a mapping layer is needed to transform the entities and relationships to different formats via the RDF representation, supported widely by different LPG databases as mentioned in Section 2. Our mapping layer generates queries using different graph query-programming languages to store and manipulate data in various graph databases.

4. Conclusion

FSEs stand to benefit from integrated knowledge in a variety of ways for providing services in an efficient manner based on up-to-date information. The described LLM-based Knowledge Graph generation based on unstructured data sources together with related customizations is promising, particularly for scaling knowledge graphs for powertrain knowledge integration. In the future, LLMs could also be utilized for automatically suggesting ontology extensions for TBox alignments with database schemas, for an initial reviewing step where LLMs act as a judge [22], as well as for enabling a chat-based user interface with translation of natural language questions to graph queries for knowledge retrieval and response. Beyond powertrain manufacturing, the proposed architecture and evaluation methodology could also be applied to further verticals involving field service engineering such as productive robotics, oil and gas, aerospace and healthcare.

Declaration on Generative AI

The author have not employed any Generative AI tools in creating the paper.

References

[1] ABB Press, ABB Ability™ Digital Powertrain for efficient, safe and reliable operations, 2019. URL: https://new.abb.com/news/detail/17846/abb-ability-digital-powertrain-for-efficient-safe-and-reliable-operations.

[2] C. P, Addressing the top 5 concerns of field service workers, 2023. URL: https://librestream.com/blog/addressing-the-top-5-concerns-of-field-service-workers/.

[3] W. Schell, P. Leoncio, Building massive knowledge graphs using automated ETL pipelines, 2024. URL: https://blog.metaphacts.com/building-massive-knowledge-graphs-using-automated-etl-pipelines.

[4] B. Stieger, M. Aleksy, Utilization of knowledge management for service business processes improvement, in: 2009 International Multiconference on Computer Science and Information Technology, 2009, pp. 171–175. doi:10.1109/IMCSIT.2009.5352730.

[5] Wikipedia, Description logic, 2025. URL: https://en.wikipedia.org/w/index.php?title=Description_logic&oldid=1283591955, page Version ID: 1283591955.

[6] World Wide Web Consortium (W3C), Resource description framework (rdf) - semantic web standards, 2014. URL: https://www.w3.org/RDF/.

[7] 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: CEUR Workshop Proceedings, Athens, Greece, 2023.

[8] Teal Labs, Inc., What is a field service engineer?, 2025. URL: https://www.tealhq.com/career-paths/field-service-engineer#:~:text=What%20does%20a%20Field%20Service,rigorous%20demands%20of%20various%20industries.

[9] ORACLE, Making sense of proactive maintenance in field service, 2021. URL: https://www.oracle.com/a/ocom/docs/field-service-proactive-maintenance.pdf.

[10] W. W. W. C. (W3C), Sparql 1.1 query language, 2013. URL: https://www.w3.org/TR/sparql11-query/.

[11] A. Parashar, What is Root Cause Analysis (RCA) in Maintenance and Why It's Important, 2025. URL: https://www.fieldcircle.com/blog/root-cause-analysis-in-maintenance/#:~:text=It's%20a%20systematic%20process%20that,intervene%20to%20prevent%20the%20failure.

[12] N. Mihindukulasooriya, S. Tiwari, C. F. Enguix, K. Lata, Text2kgbench: A benchmark for ontology-driven knowledge graph generation from text, in: International semantic web conference, Springer, 2023, pp. 247–265.

[13] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, et al., A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity, arXiv preprint arXiv:2302.04023 (2023).

[14] 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 Transactions on Information Systems 43 (2025) 1–55.

[15] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.

[16] B. Lester, R. Al-Rfou, N. Constant, The power of scale for parameter-efficient prompt tuning, arXiv preprint arXiv:2104.08691 (2021).

[17] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al., Gpt-4 technical report, arXiv preprint arXiv:2303.08774 (2023).

[18] Langchain, Llm graph transformer, 2023. URL: https://python.langchain.com/v0.2/api_

reference/experimental/graph_transformers/langchain_experimental.graph_transformers.llm.LLMGraphTransformer.html.

[19] Langchain, How to construct knowledge graphs, 2025. URL: https://python.langchain.com/docs/how_to/graph_constructing/.

[20] Neo4j, Inc., Neo4j graph database analytics, 2025. URL: https://neo4j.com/.

[21] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, J. Larson, From local to global: A graph rag approach to query-focused summarization, arXiv preprint arXiv:2404.16130 (2024).

[22] U. Simsek, E. Kärle, K. Angele, E. Huaman, J. Opdenplatz, D. Sommer, J. Umbrich, D. Fensel, A knowledge graph perspective on knowledge engineering, SN Computer Science 4 (2022) 16. doi:10.1007/s42979-022-01429-x.