An Approach to Automated Instruction Generation with Grounding Using LLMs and RAG

Laura Holvoet[1], Michael van Bekkum[1(✉)], and Aijse de Vries[2]

[1] TNO, Anna van Buerenplein 1, 2595 DA The Hague, The Netherlands
michael.vanbekkum@tno.nl
[2] TNO, Sylviusweg 71, 2333 BE Leiden, The Netherlands

Abstract. Despite ongoing digitization in industry, many companies still work with paper instructions or 'paper-on-glass' solutions (e.g., PDF files on screens). In recent years, various digital work instruction (DWI) technologies have become available that provide shop-floor employees with information during their activities, e.g., sequences of instructions for tasks at hand. Engineering new instructions in these systems for new products or product variants is however expensive and time-consuming. To scale up, there is a need for methods to generate work instructions (semi) automatically. Recently, Generative AI models and Large Language Models (LLMs) have taken center stage with their abilities to interact fluently with humans, both in understanding user questions/statements and in convincingly producing natural language texts. These models however suffer from several problems, including hallucinations where unsubstantiated content is presented as facts and lack of domain-specific data about products and procedures. For instruction generation however, we need verifiably correct statements about the task at hand. To tackle both problems, we have created a pipeline that combines the generative abilities of LLMs with explicit domain-specific data. We deploy a variant of Retrieval Augmented Generation (RAG) and incorporate an ontology that augments the instructions with additional information (policies, warnings, tools). Our results show an increase in correctness of output.

Keywords: Instruction Generation · Manufacturing · Generative AI · Large Language Models · RAG · Pipeline

1 Introduction

In a.o. manufacturing, care and construction, the European labour market is tight [1]. To help new employees work independently quicker, quality instructions are essential. Digital work instructions can have advantages over paper instructions [2], by dosing information in a clear step by step fashion, combining short texts with visual representations, e.g., pictures with annotations, videos, or

K. Alexopoulos et al. (Eds.): ESAIM 2024, LNME, pp. 224–233, 2025.
https://doi.org/10.1007/978-3-031-86489-6_23

even projections and AR [3]. Especially in a high mix low volume environment, where workers have to stay informed about the details of the current product. However, creating and keeping work-instructions up to date, requires substantial effort [4].

Information for instructions may be visual (images, videos), or text-based and stored in natural language sources (notes, manuals), or expressed in e.g., databases (parts list, available tools). This includes information on specific tools for a task, specific safety policies, or warnings ('connection is high voltage'). This knowledge is essential for the worker, but processing and interpreting these sources with often unstructured information can be difficult [5].

AI algorithms that analyze for example CAD models [4], written texts and visual data, can find relations between information in these sources and improve the quality of instructions. Large Language Models (LLMs), as a subset of Generative language models, provide the ability to interact with humans in fluent, natural language [6]. They can be used to produce natural language output upon request. By using these models as assistive technologies, instructions on work order and processes of the planned work could be extracted.

In this paper, we perform instruction generation from text input and improve correctness of the output of LLMs by using RAG and incorporating knowledge from external sources. Our contribution is threefold: (1) we apply an LLM to generate step-by-step instructions from unstructured instruction text, (2) we use an ontology with the main concepts of the assembly process that provides context information to generated instructions, (3) we use this ontology to augment the instructions generated by the LLM. We have demonstrated our setup in a small-scale demonstration scenario for a single assembly.

2 Related Work

Digital Instructions. Research has so far focused on reducing the time to adapt to changing demands, or improved product designs by automating the creation of manufacturing instructions [7]. To that end, information can be extracted from e.g. CAD models [4,8,9] or e.g. from workload models [10] to create instructions for an assembly sequence. AI algorithms can also be applied to generate digital, interpretable instructions [11].

Large Language Models. Language modeling gained popularity recently, due to the emergence of Large Language Models (LLMs) [12]. These are pre-trained language models of a certain size (potentially billions of parameters), demonstrating better performance than smaller-scale LMs and capabilities that emerge from their size [12,13]. Thanks to their natural language generation capabilities, their use in various fields of industry is increasing [12,14]. There are however numerous challenges related to their application [15], such as: hallucinations [16], lack of interpretability [17], lack of domain specific knowledge [12], high training costs [15].

Ontologies are structured models, that explicitly describe knowledge and generate interpretable results due to their symbolic reasoning capacity [18]. They can be used to store schemas for instructions and/or instructions themselves [19]. Ontologies can easily be updated to include new concepts and data, whereas LLMs require costly fine-tuning or retraining. Ontologies however, require much more expertise on explicating domain knowledge than LLMs [20].

Integration of LLMs and Ontologies can leverage the advantages of both because of their complementary nature and addresses some of the issues associated with LLMs as described above. Pan et al., [17] present a research overview and a roadmap for integration of LLMs and Knowledge Graphs.

Retrieval Augmented Generation or (RAG) ([21]) is a technique for retrieving information from an external database in order to ground the answers of Large Language Models and to enhance their trustworthiness, accuracy, and reliability [22]. This technique helps to solve some of the common issues regarding LLMs, such as hallucinations, lack of domain specific knowledge and knowledge cut offs.

3 Method

This section will provide a description of the pipeline created during our research and an overview of the knowledge base that models the assembly process.

3.1 Pipeline Structure

In this work, we implement a variant of Retrieval Augmented Generation and apply it to instruction generation. In a naive RAG architecture (Fig. 1), the documents that constitute the external knowledge are split into chunks, then a numerical embedding of each chunk is created and stored in a vector database. When the LLM is queried, the query embeddings are used to look up similar documents in the vector database. The documents most similar to the query are then used to augment the model's answer. While this approach works well with unstructured information, we observed that it does not have good results in applications that require high retrieval accuracy. Therefore, we propose a pipeline that uses structured data (i.e. an ontology) as external knowledge and inverts the retrieval and the generation steps compared to the traditional RAG architecture. This has proven to work better than the traditional RAG architecture, as mentioned in Sect. 4.

The pipeline, shown in Fig. 2, is made up of two main components: the Large Language Model, for instruction generation and the Knowledge Retriever, responsible for retrieving the relevant context information.

The Large Language model is used to extract short step by step instructions from snippets of unstructured (spoken) instruction text. Every time it is prompted, the language model is given an example of the expected structure of

the output (the same example for every prompt) and is asked to generate short step by step instructions using the provided text snippet.

After the instruction generation step, the instructions are passed to the knowledge retriever. The function of the knowledge retriever is to identify the action a specific instruction is referring to and retrieving all the relevant information about it from the ontology. The current action is identified by querying a vector database. By splitting the text into snippets that describe a single step, we ensure that each vector in the database represents a step and holds information about the action carried out in that step. Via a similarity search with the instruction, the current action is returned, and used to retrieve the relevant information from the ontology.

Finally, the knowledge retrieved from the ontology is added to the instructions generated by the large language model. This is done by simply 'appending' the additional information to the generated instruction text. In Sect. 4 we show and discuss examples of the input and output of the pipeline and its components.

3.2 Ontology Creation

The assembly process is represented in an ontology (knowledge base), which contains some of the main concepts of the assembly process: steps, actions, components and tools and the relationships between these concepts, e.g.: a step consists of an action; an action requires a tool etc. The structure of the ontology is inspired by and simplified from existing ontologies in the manufacturing process domain [23,24] and can be seen in Fig. 3. The ontology can be filled with instances of each of the concepts, based on the specific application. In our application, the component class contains different types of components, such as a tire, a rivet, a bolt etc.

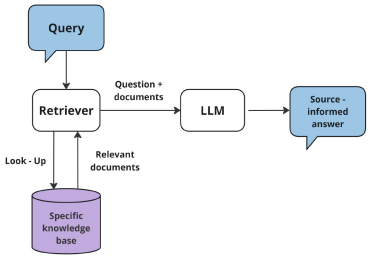

Fig. 1. Typical naive RAG architecture.

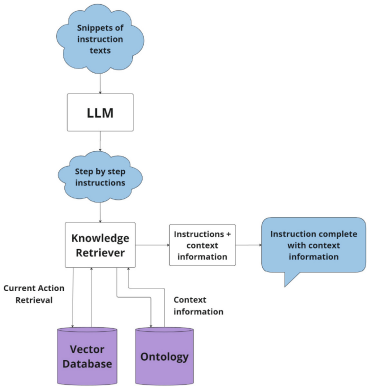

Fig. 2. Architecture of the pipeline.

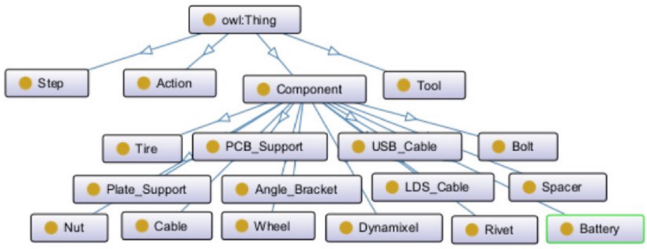

Fig. 3. Structure of the ontology (extracted from Protégé [25]).

4 Experiment

In this section, we will provide a description of the experiments that were carried out, with snippets of input and output results.

4.1 Method

To test our pipeline, we have created an example implementation using the TurtleBot Burger Robot [26] assembly instructions as our dataset. The input data was created by writing an unstructured text that contains assembly instructions (simulating instructions that could be extracted from an instruction video). The assembly process was then modelled using the ontology structure in Sect. 3.2. An important requirement for this pipeline was the use of an open source LLM, because it would allow the model to be run locally and it would not put any private company data at risk. Therefore the model that was chosen was a fine-tuned model based on Llama2 [27], called Xwin-LM [28]. To minimize memory consumption and inference time, a quantized [29] version of this model was used, that is 7 times lighter than the not quantized version. The vector database was created using Pinecone [30] and a sentence transformer [31] model was used to create the vector embeddings. We expect that this pipeline to generate instructions will yield the following results:

1. It is expected that the LLM will generate brief and concise instructions by paraphrasing the input text and by excluding the phrases present in spoken text, but unrelated to the assembly process.
2. The external source of information (i.e., the ontology) is expected to improve the instructions generated by the LLM, by augmenting the LLM's answer with knowledge that it would otherwise not have access to.
3. We expect that inverting the retrieval and generation steps will improve the retrieval step.
4. We expect that an ontology will be a more advantageous way of representing background knowledge specific to this application (i.e., list of tools and components required for an action, warnings etc.) rather than unstructured text.

The pipeline was tested using the Turtle Bot dataset. Snippets of the unstructured instruction text are used as input to the pipeline. First, the Large Language model is prompted to generate brief step-by-step instructions based on the input text. Then the generated instructions are passed to the knowledge retriever. For each step, the retriever finds the action the instruction is referring to and retrieves all the information related to it from the ontology. In this case the concepts related to an action are: required components, required tools and warnings. Finally, each assembly instruction is combined with the related information retrieved in the previous step and the final output is obtained.

4.2 Result

An example of the experiments described above can be seen in Fig. 4. They illustrate the input to the pipeline, the intermediate output of the LLM and the final output, that incorporates the information from the ontology. After the experiments, it was observed that:

1. Given the input text (Fig. 4), the model outputs concise instructions, that follow a stepwise format, therefore succeeding in reducing unnecessary text.
2. Compared to a scenario where the LLM is queried without an external source of information, this pipeline allows to augment the output by adding any amount of new information to the output. The ontology can be easily updated to add new information about an assembly process (updated tools, warnings, or protocols) as well as to remove any outdated knowledge without needing to retrain the language model.
3. Inverting the traditional RAG structure proved to work well. The reason for this is that the input is made up of short paragraphs of unstructured text that do not contain a predefined number of assembly steps. This makes it impossible to determine *a priori* how many steps are referred to in one paragraph, therefore how much and what information should be retrieved. Generating the instructions beforehand makes it easier to 'separate' the text into steps and to query the ontology to obtain information about each step.
4. The ontology is better suited to represent this background information. The background information used for our application consists of lists of components or tools, making it better suited for a structured representation. Retrieving all the components associated to an action becomes easier than extracting this type of information from text.
5. The LLM still has the tendency to hallucinate. The hallucinations appear when the model, after correctly generating the instructions, continues generating text until it reaches the token limit. The hallucinations were still pertinent to the instructions but were not correct and were not generated using the provided input. This most likely is due to the model size (as mentioned, the model is a quantized version of a 13 billion parameter model).
   We have attempted to solve the hallucination problem using prompt engineering and parameter tuning. Few shot examples were used to show the model the expected output. Giving a one-shot example to the model helped improve

the results significantly and the model was able to adhere to the given format. However, providing more than one example in the prompt did not lead to the expected improvement (i.e., that the model would learn when to stop the generation).
Parameter tuning, i.e., lowering the temperature parameter or decreasing the token limit, did not lead to significant improvements.
Finally, we attempted to use a different LLM for the task [32]. It showed promising results, however due to time constraints, we were not able to thoroughly test it within the pipeline.

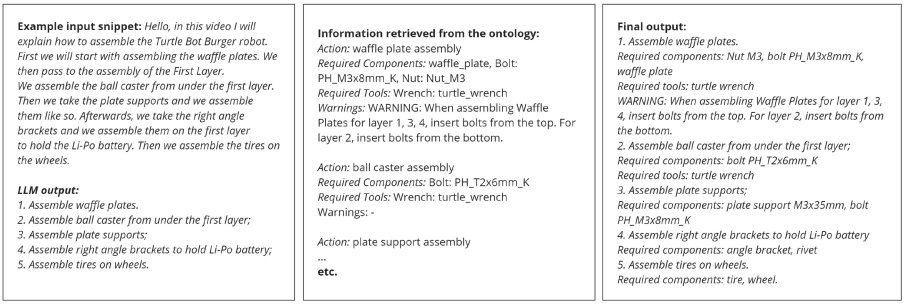

Fig. 4. Examples of the input and output of the pipeline components.

5 Conclusion

LLMs are a promising technology that can support in producing convincing natural language instruction texts and alleviate the burden of manual engineering. In this paper, we have created a pipeline that mitigates some of the unwanted hallucination effects of LLMs by applying RAG. We show how an ontology of explicit domain-specific instruction data supports the LLM by providing information the LLM does not possess. More specifically, we state that:

LLMs Improve over Manual Engineering Instructions: The pipeline only requires some (spoken) text from a person that explains how to do the assembly and the one-off cost of creating the ontology.

Advantages of Using the Ontology: An ontology adds structured, external information that the LLM does not possess, thus improving the quality of the generated instructions over using a simple text source;

Hallucinations Remain: The LLM still starts hallucinating after correctly generating the instructions. Larger models could potentially reduce hallucinations. In addition, a language model could be fine-tuned specifically for this task.

Manual Engineering in the Ontology Remains: The ontology and all the steps, actions and tools need to be created and inserted manually, respecting the original structure of the ontology. An automated approach to ontology creation could further reduce the manual labor.

In order to evaluate the quality of answers generated by the LLM and its ability to generate complete and comprehensible instructions, we propose to obtain evaluations by utilizing both human evaluators and existing LLM based validators as a future direction [33]. The work presented in this paper can also be a solid foundation for other types of applications, such as a Q&A system for instructions on specific procedures (e.g., for maintenance purposes). Our current, limited setup is therefor a first step towards a more elaborate investigation of our approach in a more dynamic, real-world manufacturing environment, with a.o. challenges of far bigger datasets.

Acknowledgement. This project received a contribution from the Growth Fund programme NxtGen Hightech.

References

1. Eurofound: The changing structure of employment in the EU: annual review (2023)
2. Musen, M.A.: Should firms use digital work instructions? - individual learning in an agile manufacturing setting. J. Oper. Manag. **68**(1), 94–109 (2022)
3. Kablan, Z., Erden, M.: Instructional efficiency of integrated and separated text with animated presentations in computer-based science instruction. Comput. Educ. **51**, 660–668 (2008)
4. Gors, D., Put, J., Vanherle, B., Witters, M., Luyten, K.: Semi-automatic extraction of digital work instructions from cad models, vol. 97, pp. 39–44, Elsevier B.V. (2020)
5. Leoni, L., Ardolino, M., El Baz, J., Gueli, G., Bacchetti, A.: The mediating role of knowledge management processes in the effective use of artificial intelligence in manufacturing firms. Int. J. Oper. Prod. Manage. **42**(13), 411–437 (2022)
6. Hammond, K., Leake, D.: Large language models need symbolic AI. In: CEUR Workshop Proceedings, vol. 3432, pp. 204–209 (2023)
7. Jain, N.K., Jain, V.K.: Computer aided process planning for agile manufacturing environment (2001)
8. Zogopoulos, V., Geurts, E., Gors, D., Kauffmann, S.: Authoring tool for automatic generation of augmented reality instruction sequence for manual operations. Procedia CIRP **106**, 84–89 (2022)
9. Koga, Y., Kerrick, H., Chitta, S.: On CAD informed adaptive robotic assembly (2022)
10. Evangelou, G., Dimitropoulos, G., Michalos, G., Makris, S.: An approach for task and action planning in human-robot collaborative cells using AI. Procedia CIRP **97**, 476–481 (2021)
11. Grappiolo, C., Pruim, R., Faeth, M., Heer, P.D.: ViTroVo: in vitro assembly search for in vivo adaptive operator guidance an artificial intelligence framework for highly customised manufacturing, pp. 3873–3893 (2021)
12. Zhao, W.X., et al.: A survey of large language models (2023)

13. Wei, J., et al.: Emergent abilities of large language models (2022)
14. Chang, Y., et al.: A survey on evaluation of large language models (2023)
15. Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., McHardy, R.: Challenges and applications of large language models (2023)
16. Tonmoy, S.M., et al.: A comprehensive survey of hallucination mitigation techniques in large language models (2024)
17. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., Wu, X.: Unifying large language models and knowledge graphs: a roadmap. IEEE Trans. Knowl. Data Eng. **36**, 1–20 (2024)
18. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. **5**(2), 199–220 (1993)
19. Woods, C., French, T., Melinda, H., Bikaun, T.: An ontology for maintenance procedure documentation. Appl. Ontol. **18**, 169–206 (2023)
20. de Almeida Falbo, R.: SABiO: systematic approach for building ontologies. In: ONTO.COM/ODISE@FOIS (2014)
21. Lewis, P., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks (2021)
22. Gao, Y., et al.: Retrieval-augmented generation for large language models: a survey (2024)
23. Lemaignan, S., Siadat, A., Dantan, J.Y., Semenenko, A.: MASON: a proposal for an ontology of manufacturing domain. In: Proceedings - DIS 2006: IEEE Workshop on Distributed Intelligent Systems - Collective Intelligence and Its Applications, pp. 195–200 (2006)
24. Cao, Q., Zanni-Merk, C., Reich, C.: Ontologies for manufacturing process modeling: a survey, June 2018
25. Musen, M.A.: The protégé project: a look back and a look forward. AI Matters **1**(4), 4–12 (2015)
26. MS Windows NT kernel description. https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/. Accessed 30 Oct 2023
27. Touvron, H., et al.: Llama 2: open foundation and fine-tuned chat models (2023)
28. X.-L. Team: Xwin-LM, September 2023
29. Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D.: GPTQ: accurate post-training quantization for generative pre-trained transformers (2023)
30. Pinecone Systems: Pinecone (2022). Accessed 01 Nov 2023
31. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, November 2019
32. Jiang, A.Q., et al.: Mixtral of experts (2024)
33. Shankar, S., Zamfirescu-Pereira, J.D., Hartmann, B., Parameswaran, A.G., Arawjo, I.: Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences, April 2024