Incremental Learning for Knowledge-Grounded Dialogue Systems in Industrial Scenarios

Izaskun Fernandez[1], Cristina Aceta [1], Cristina Fernandez [1], María Inés Torres [2], Aitor Etxalar[2], Ariane Mendez [3], Maia Agirre [3], Manuel Torralbo [3], Arantza del Pozo [3], Joseba Andoni Agirre[4], Egoitz Artetxe [4], Iker Altuna[4]

[1]TEKNIKER - Basque Research and Technology Alliance (BRTA),
[2] Speech Interactive Research Group (SPIN) - University of the Basque Country (UPV/EHU),
[3]Vicomtech Foundation - Basque Research and Technology Alliance (BRTA), [4]IMH Campus

Correspondence: izaskun.fernandez@tekniker.es

Abstract

In today's industrial landscape, seamless collaboration between humans and machines is essential and requires a shared knowledge of the operational domain. In this framework, the technical knowledge for operator assistance has traditionally been derived from static sources such as technical documents. However, experienced operators hold invaluable know-how that can significantly contribute to support other operators. This work focuses on enhancing the operator assistance tasks in the manufacturing industry by leveraging spoken natural language interaction. More specifically, a Human-in-the-Loop (HIL) incremental learning approach is proposed to integrate this expertise into a domain knowledge graph (KG) dynamically, along with the use of in-context learning for Large Language Models (LLMs) to benefit other capabilities of the system. Preliminary results of the experimentation carried out in an industrial scenario, where the graph size was increased in a 25%, demonstrate that the incremental enhancing of the KG benefits the dialogue system's performance.

1 Introduction

Human-Machine Interaction (HMI) is revolutionizing traditional industrial processes. Smart manufacturing relies on the collaboration between highly advanced machinery and the knowledge and decision-making abilities of human operators. The industry of the near future requires qualified personnel specialized in technologies such as robotics and artificial intelligence (AI), capable of making informed decisions based on these factors. In this context, a human-centered approach positions operators as a crucial element in new industrial plants. Thanks to the latest technological advances, voice interaction between operators and industrial manufacturing systems or machines is now feasible. Moreover, these technologies are *hands-free* and *eyes-free*, enabling operators to perform physical tasks, *support natural language communication* that requires minimal training, and are *highly flexible*, allowing communication at various levels of detail. Consequently, there has been an increase in the number of prototypes and systems exploring the use of voice as a natural interaction interface between operators and machines in industrial environments in recent years. Additionally, dialogue modeling and management have drastically changed due to the recent success of large language models (LLMs). However, any application based on LLMs needs reliable and up-to-date knowledge sources. In particular, industrial scenarios require robust models capable of handling very technical and precise knowledge, which is necessary for tasks shared by humans and machines.

Traditionally, HMI has relied on rule-based systems to represent knowledge and actions, ensuring everything remains under control. As a result, these interaction systems are static, failing to capture the expert human knowledge of the factory that is not documented or included in the system's knowledge base. This limitation can be addressed through the concept of Human in the Loop (HIL), also known as Operator in the Loop (OIL) in industrial contexts. These AI systems facilitate collaboration between humans and machines to enhance results and accelerate the learning process. The HIL paradigm involves continuous interaction throughout all post-deployment stages of AI models. As illustrated in Figure 1, in the industrial sector, the OIL paradigm enables the integration of expert knowledge into HMI interfaces by providing feedback using natural language. This approach allows voice interaction systems to evolve over time, adapting to the unique dynamics of each factory and incorporating the expertise that operators develop.

In this work, an OIL incremental learning approach to manage knowledge-grounded, task-oriented dialogue (TOD) systems in industrial set-

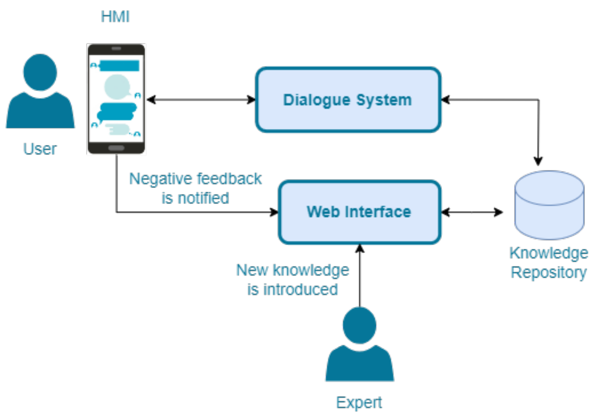


Figure 1: Operator in the Loop paradigm.

tings is proposed, being its main contributions (1) extending a previously defined ontology to support the management and storage of new knowledge provided by experts; (2) developing online learning capabilities to collect user feedback, thereby updating and expanding a knowledge graph and (3) developing an LLM-based natural language understanding (NLU) system that queries a KG to constrain it within the task. Preliminary evaluations show promising results in NLU performance and KG grounding.

The rest of the paper is structured as follows: Section 2 presents the related work. The proposed knowledge graph-based incremental dialogue system with the detailed description of each of its modules is described in Section 3, and the initial validation results in Section 4. Finally, conclusions of this work are shown in Section 5.

2 Related work

The current state of the art considers knowledge graphs as a useful asset in industrial settings and, more specifically in human-centric approaches (Abonyi et al., 2024), such as robot interaction and collaborative manufacturing (Nagy et al., 2024). In this line, approaches such as the one proposed by Nagy et al. (2024) are observed, in which knowledge graphs are used to model factors related to the operator and their conditions, such as movements or collaboration with machines. Moreover, knowledge graphs have been used in this scenario for task-oriented dialogue, which enable operators to communicate to industrial systems in a more natural way. In this context, knowledge graphs have been traditionally used to model the domain of the use case, providing a detailed representation of the scenario and reducing ambiguity between the agents involved (Sidi Yakoub et al., 2015). However, more modern approaches also make use of knowledge graphs for dialogue management (Teixeira et al., 2021; Aceta et al., 2022)

Of course, this process also has an impact on dialogue management, since one of the most widespread techniques is to obtain this information from users. To do this, dialogues are generated dynamically to be able to obtain the necessary information for the system to learn, as well as the appropriate moment for it, based on a strategy (Liu and Mazumder, 2021). Some approaches also base these interactions on the feedback obtained from the user taking into account, for example, evaluations such as “it’s not what I wanted” or “you didn’t understand me well” (Veron et al., 2021).

In the field of Natural Language Processing (NLP), there’s a clear surge in leveraging state-of-the-art strategies across multiple applications, particularly through the deployment of pre-trained Large Language Models (LLMs) in dialogue systems. Ozdemir (2023) describes these models as AI models that often, though not exclusively, stem from the Transformer architecture. They are crafted to understand and generate human language, code, and beyond. Also, they are trained on immense troves of text, and they can tackle a vast array of language-related tasks, from simple text classification to elaborate text generation. As highlighted by this author, the LLMs available in the market (like various versions of GPT, Gemini, Llama, among others) have been pre-trained on extensive datasets from diverse sources using distinct methodologies. Thus, not all LLMs perform equally, and their training processes significantly influence their performance in specific applications.

Therefore, to optimize pre-trained language models for task-oriented dialogue systems, different works employ models like Alpaca, GPT-Neo, BART, T5, Llama 2 and GPT-3.5 (Hudeček and Dušek, 2023; Andreas et al., 2022; Li et al., 2022; Hu et al., 2024); among others. Also, different authors adopt various approaches for constructing these dialogue systems. Prominent among these is fine-tuning pre-trained language models using methods like LoRA (Low Rank Adaptation) (Andreas et al., 2022; Li et al., 2022), prompt tuning (Cao, 2023; Hudeček and Dušek, 2023) and Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022), among others. This indicates a clear trend in using LLMs in TOD systems and various optimization strategies. However, many of these methods demand substantial specific data for training as they are data-driven, which may not be available for certain industrial use cases.

To address this issue and avoid the need for handcrafted rules, in-context learning approaches are becoming increasingly popular. These approaches involve designing prompts using snippets of example dialogues, the user's goal, and the dialogue history (Sekulić et al., 2024). This optimization method, known as prompt tuning, allows adapting the model to task requirements without requiring a corpus or extensive training, just relying on natural language instructions to guide the model's behaviour.

3 Knowledge-Grounded Incremental Task-oriented Dialogue System

Two of the most common applications of TOD systems in industrial scenarios are to provide assistance through processes and to deliver tasks to a certain industrial intelligent system. Therefore, the expected interactions from the user can mainly be classified as navigation instructions through processes and action requests to industrial systems, respectively. The TOD system's responses, on the other side, must be in the form of steps of the processes on which the user will request assistance for the former, and the corresponding machine-readable action for the latter.

So, in this type of scenarios, towards an incremental approach, feedback may be useful in these two situations, mainly: (1) the content presented does not meet the needs of the user or (2) the interpretation of the interaction indicates that what the user wants to do next or deliver to the system is not appropriate.

This work presents the extension and adaptation of KIDE4I, presented in Aceta et al. (2022) and based on the TODO Ontology (Aceta et al., 2021), to provide it with feedback-capturing and management capabilities. The aim of such task is to achieve a system that is capable of learning from interactions with users over time and, thus, improve its interpretation and dialogue capacities, as well as adapting to the users' needs. To this end, the following aspects have been addressed:

1. Extension of the TODO ontology to support the management and storage of new knowledge based on feedback (described in Section 3.1).
2. New functionalities to generate dialogues aimed at collecting feedback and to update knowledge extracted from it (described in Section 3.2) .

Likewise, and towards assessing the benefits when updating KIDE4I with the most recent technologies, in-context learning (ICL) of LLMs through prompt-tuning has been explored and implemented in the natural language understanding (NLU) module, as detailed in Section 3.3. This task has allowed to compare more traditional strategies, such as rule-based ones, with the most disruptive one nowadays: the use of LLMs in scenarios with limited resources (in terms of training corpus), such as industrial ones.

3.1 Industrial-Assistance-Oriented Incremental Knowledge Graph

By definition, a knowledge graph focuses on representing relationships and capturing real-world connections, ideally based on an ontology that provides the formal framework for defining the terms and concepts used in that representation.

As described previously, the focus of this work is developing a knowledge-graph-based TOD system for industrial scenarios which is based on technical documentation and expert knowledge that can be extended over time through, for instance, feedback gathering. To achieve such a system it is necessary to construct a knowledge graph that formally represents all this information, relying on a agreed ontology that allows an incremental learning approach.

The core ontology for developing the knowledge graph in the context of this work is TODO (Aceta et al., 2021), the main modules of which can be seen in Figure 2. This modular ontology is designed to enable task-oriented dialogue systems to interact naturally with users at both understanding and communication levels by distinguishing two main areas of knowledge: domain (TODODom) and dialogue (TODODial), respectively. It can be readily adapted to various industrial settings, thus minimizing the time and cost of adaptation. Additionally, it supports the storage and reproduction of the dialogue process, allowing for learning from new interactions. However, this tracing capability, although being a good starting point for supporting an incremental learning approach, does not support the generation and management of user feedback. In order to solve this gap, the TODO ontology has been extended.

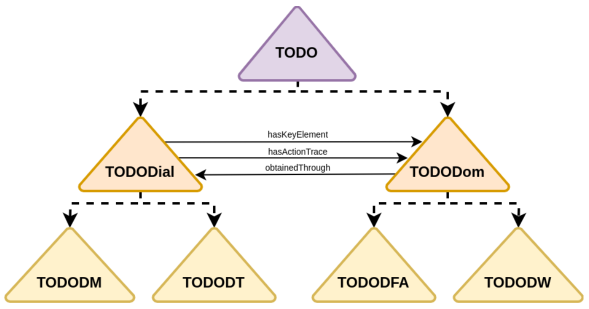


Figure 2: TODO ontology (Aceta et al., 2021)

In that extension and adaptation task, new classes and relationships that allow representing the key concepts aimed at collecting feedback have been added. More precisely, 2 classes (C) and 4 object properties (OP) have been created in the TODODom module and 2 classes in TODODM, which are listed below, by module.

TODODom (domain)

- DefinitiveLexicalUnit (C), to depict lexical units (i.e., variants) that have not been added through feedback (i.e. manually or in a supervised way) or lexical units that have been added through feedback several times.

- ProvisionalLexicalUnit (C), to depict lexical units that have been added through feedback but the confidence to consider them as definitive is still low.

- hasDefinitiveLexicalUnit (OP), to relate frame heads (i.e., generic terms to agglutinate different variants) to their corresponding definitive lexical units.

- hasProvisionalLexicalUnit (OP), to relate frame heads to their corresponding provisional lexical units.

- isDefinitiveLexicalUnitOf (OP). Inverse property of hasDefinitiveLexicalUnit.

- isProvisionalLexicalUnitOf (OP). Inverse property of hasProvisionalLexicalUnit.

TODODM (dialogue management)

- NewLexicalUnitConfirmationRequest (C), to request the user for confirmation to relate a lexical unit to a specific frame head.

- ActionDetectedResponse (C), to inform the user that it has detected an action (for which the command includes a new reference to potentially be added to the graph).

These classes and properties have been added by following the LOT methodology (Poveda-Villalón et al., 2019), which makes sure that knowledge is modelled into the ontology ensuring its quality. Therefore, the quality of the ontology (compared to the results obtained in Aceta et al. (2021)), has not been affected.

With the ontology ready, a manual instantiation of the newly-modelled, dialogue-related classes has been carried out, in order to offer the dialogue manager variations to interact with the user and direct the dialogue to capture feedback, such as "Can you confirm that `{item}` is a related word?". The rest of the dialogue-related instantiations have been reused from the generic instantiation of TODODial (Aceta et al., 2022).

As for the domain section of the knowledge graph, it is instantiated automatically. First of all, the relevant procedures have been defined by the experts by using an interface designed to simplify the instantiation process. In a nutshell, this interface, once a procedure is defined, generates, first of all, a JSON file. This JSON file, by following an *Extract, transform and load* (ETL) process, is transformed into RDF and uploaded to the RDF store, which, in this case, has been Virtuoso 8.3. An example snippet of an instantiated procedure can be found in Appendix A.

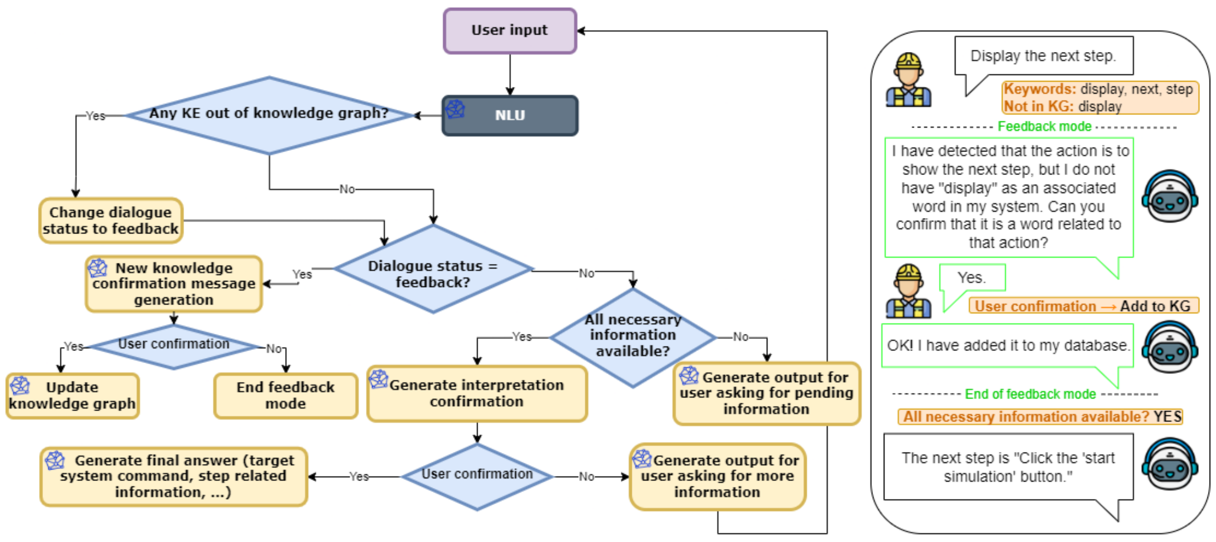
This first graph version enables the system to be ready to be used and its knowledge to be extended through user feedback in subsequent interactions.

3.2 Dialogue Management Supporting Incremental Approach

Once the ontology is extended and the dialogue instances for collecting feedback are ready, as reported in Section 3.1, it is necessary to add to the dialogue manager the capability to extract the new knowledge to be included in the system.

As described previously, the two situations that may require feedback gathering would be when the system is not capable of correctly interpreting an user request and when the information provided by the system is not accurate.

To respond to the first situation, the dialogue manager has been extended so that, instead of asking the user to reformulate the request because they


User input
NLU
Any KE out of knowledge graph?
Yes
No
Change dialogue status to feedback
Dialogue status = feedback?
New knowledge confirmation message generation
All necessary information available?
User confirmation
Update knowledge graph
End feedback mode
Generate interpretation confirmation
Generate output for user asking for pending information
Generate final answer (target system command, step related information, ...)
Generate output for user asking for more information
Keywords: display, next, step
Not in KG: display
Feedback mode
User confirmation → Add to KG
End of feedback mode
All necessary information available? YES

Display the next step.

I have detected that the action is to show the next step, but I do not have "display" as an associated word in my system. Can you confirm that it is a word related to that action?

Yes.

OK! I have added it to my database.

The next step is "Click the 'start simulation' button."

Figure 3: System workflow, including feedback management (left), along with an interaction example that requires feedback and follows the aforementioned flow (right).

are not able to understand it, the system enters *feedback mode*. The main goal of this mode is that the system is able to link new key elements to an action or action slot in subsequent interactions. For that, a clarifying request for the user, as a question, is triggered, in an intent to link the key element(s) extracted from the interpretation module with some of the classes/instances of the semantic repository. When the user responds to said system request, it is interpreted and the dialogue status is checked. If the status is feedback mode, once the user confirms the interpretation, the system launches a request to update the knowledge graph. This update, which has been automated by developing a REST API service, represents the extension of the base knowledge of the system. However, since it is an automated process, and to achieve controlled growth, this new knowledge is marked as obtained from feedback in the base (*provisional*, in accordance with what has been established in the adaptation of the TODO ontology, depicted in Section 3.1). Figure 3 visually summarizes the system's dialogue flow, with the new feedback management capabilities to learn based on interactions with the user and update knowledge dynamically.

When it comes to the second case, in which the user's disapproval of a system response is due to the fact that the content does not cover their needs, this feedback must trigger an action by an expert to review the system's knowledge and update it if appropriate. For this case, a graphical interface has been developed so that it enables the user to indicate their disagreement with the content and the expert to edit the content of the processes described in the repository when necessary. By the time this edition occurs, a functionality has been developed in the dialogue manager, which allows updating the knowledge graph with the new content. This new revised and improved data is what the system will use onwards as part of the extended knowledge graph.

3.3 Natural Language Understanding

The functions of the Natural Language Understanding (NLU) component are, first, to determine if a transcribed user voice command is classified as a polar interaction (e.g. "yes", "no"). If it is, it is in charge of determining whether it is positive or negative. If no polarity is detected, a key element extraction (KEE) component is raised to extract the relevant information from the command, as shown in Figure 4.

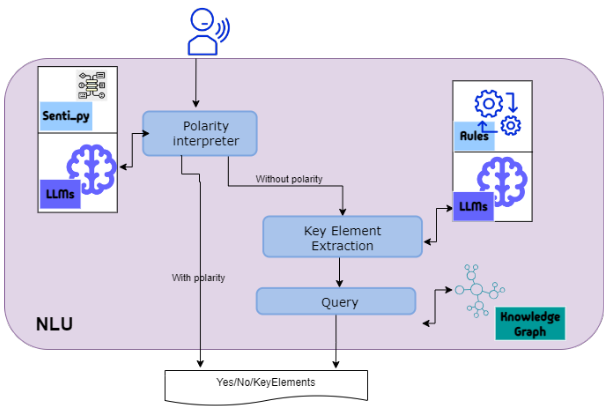


Figure 4: NLU pipeline

For the present work, two different approaches have been tested for the polarity interpreter. The first relies on the KIDE4I implementation, using a sentiment analysis algorithm. As for the second, an LLM-based approach has been implemented. More specifically, the GPT-3.5 Turbo model has been adapted through the in-context learning prompting strategy.

For the rest of interactions, namely non-polarized interactions, the KEE module intervenes fulfilling a slot-filling task. Similarly to the polarity interpreter, two different implementations have been used. For the first, again, the KIDE4I rule-based approach has been followed, whereas, for the second, the same LLM approach as above has been used. In this case, within the prompt, its function and the slots to be filled are indicated.

After detecting the slot values, it is essential to verify that these values correspond to recognizable world elements within the target system. This process involves querying the knowledge graph and comparing the detected slot values with those stored in the graph. If the key elements identified by the NLU are found, they are retained in the component's final result. Otherwise, the unrecognized values are discarded, asking for the missing information later on.

4 Initial Results

The extended TOD system with feedback capabilities has been tested and validated in an experimentation task. In this particular case, the system's function was to provide support through the different phases for manufacturing a piece using CNC programming on a milling machine. When given instructions, the users that were not satisfied with the answer given by the assistant would mark the response as inadequate. Some initial results related to this experimentation, mainly regarding the impact of those iterations in the knowledge graph, are presented in Section 4.1.

In terms of incrementing knowledge through feedback, the assistance scenario is suitable for evaluation. However, the variability in the interactions (e.g. "Show me the next step", "I need more information") is limited. Due to this, and in order to provide more insights, the LLM-based and rule-based NLU components have been tested in a collaborative bin-picking scenario, which is richer in terms of references to key elements in user commands. The obtained results for both have been compared in order to determine if the LLM-based approach, which makes the task of adapting the dialogue system easier, is able to maintain or improve the accuracy of the rule-based approach. The experimental setup, as well as the benchmark results, are presented in Section 4.2.

4.1 Incremental TOD System in Use

The proposed validation scenario, as noted previously, is about CNC programming to manufacture a piece on an IKASMAK 5.1 milling machine. To do so, 15 users were requested to be assisted by the assistant described in Section 3 and to give insights about different procedures along the different phases of the programming process upon a given user request. The language used in this scenario was Spanish.

Furthermore, while the user is interacting with the system, if it does not present the desired information or does not perform as expected, the user can vote negatively the answer. This vote triggers a review alert for the expert, who will review the dialogue flow and, if necessary, update the knowledge graph to try to solve the gap, as shown in Figure 5.

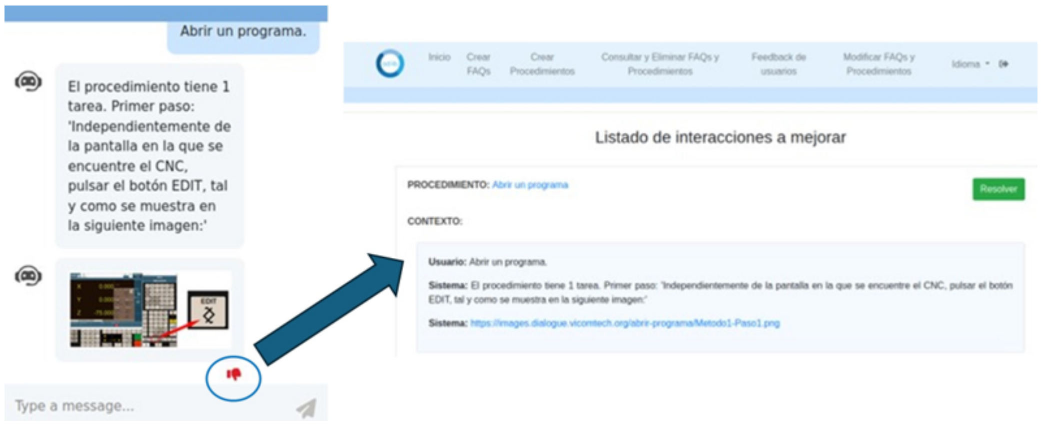
This expert review, then, improves and, some times, even increases the information in the knowledge graph, and so, the accuracy of the system in further uses. The following subsection shows the evolution of the graph after the experimentation where, at some point within the dialogues, 22 of the total of 551 turns were marked as the response from the system was not valid, which triggered an expert review. Although it could sound like a high number, the time required for the review and update of the knowledge graph has not exceeded 5 hours, a process that would have taken much longer if done through other methods (e.g. manual instantiation) and would have required an ontology expert to perform it.

4.1.1 Impact on the Knowledge Graph and Initial Analysis

So as to show the evolution of the graph before and after the expert review process, Tables 1 and 2, respectively, are depicted below.

In the case of Table 1, the average and total number of the different instances for each relevant class in procedure definition can be seen, out of a total of 341 total instances.

Thanks to the user feedback and expert review process, and as it can be seen in Table 2, the number of total instances has increased. More specifically,


Abrir un programa.

El procedimiento tiene 1 tarea. Primer paso: 'Independientemente de la pantalla en la que se encuentre el CNC, pulsar el botón EDIT, tal y como se muestra en la siguiente imagen:'

Type a message...

Listado de interacciones a mejorar

PROCEDIMIENTO: Abrir un programa

CONTEXTO:

Usuario: Abrir un programa.

Sistema: El procedimiento tiene 1 tarea. Primer paso: 'Independientemente de la pantalla en la que se encuentre el CNC, pulsar el botón EDIT, tal y como se muestra en la siguiente imagen:'

Sistema: https://images.dialogue.vicomtech.org/abrir-programa/Metodo1-Paso1.png

Figure 5: User discontent triggering expert review requirement

Proc	**AP**	**P**	**Met**	**Task**	**Step**	**PSI**
10	14.9	1	1.3	1.3	6.9	8.7
Total	149	10	13	13	69	87

Table 1: Number of procedures ("Proc") and average and total number of activation phrases ("AP"), procedures ("P"), methods ("Met"), tasks, steps and additional information ("PSI"), in the moment of the experimentation.

Proc	**AP**	**P**	**Met**	**Task**	**Step**	**PSI**
12	14.6	1	1.6	1.6	8	9.2
Total	175	12	19	19	96	110

Table 2: Number of procedures ("Proc") and average and total number of activation phrases ("AP"), procedures ("P"), methods ("Met"), tasks, steps and additional information ("PSI"), after the user feedback and expert review.

90 more instances have been added, making a total of 431 (that is, a 25% more knowledge). Among these instances, new activation phrases have been added for the existing procedures and, furthermore, two new procedures have been included: "Detener un programa" ("Stop a program") and "Configurar el avance" ("Configure the advance"). These two procedures have been added following the same format as the rest of procedures, an example of which can be found in Appendix A.

4.2 Natural Language Understanding: Benchmarking

The scenario used for the NLU component validation is a classification task, in which a bin-picking collaborative robot is able to classify cartridges by depositing them in different boxes, according to user commands in Spanish. More specifically, the robot can pick up different ink cartridges from a table, identify their color and brand, and sort them into two separate containers, based on the operator's instructions. The operator must use natural communication to inform the robot about the type of cartridge and the designated box. This communication involves not only voice commands but also gestures to indicate the destination. Consequently, the key element extraction module must identify actions and targets related to brands, colors (of the cartridges), and containers. Additionally, it must detect references to gestures indicated by phrases like "here" or "this", which enhance the verbal instructions and provide supplementary information.

4.2.1 In-context Learning LLM vs Rule-based

In order to evaluate the behaviour of NLU in the different systems (rule-based and LLM-based), similar dialogues have been established with the same start of dialogue and the same end goal. In this way, they can be compared in number of turns and the performance of the NLU can be analysed. Therefore, based on these dialogues, the results of the KEE have been analysed for each turn.

A total of 74 dialogues were established with a total of 12 different users. However, the number of dialogue turns (159 for the rule-based and 176 for the LLM-based) and total KEE module intervention (130 and 108, respectively) varies due to the structure of the dialogue –which is slightly different for each system– and the performance of the different modules. The performance of both approaches can be observed in Table 3. In order to have a better approximation of the results, they have been classified between "fully detected", "partially detected" and "not detected" to refer to when

	Fully detected		Partially detected		Not detected		Out-of-scope errors	
	%	#	%	#	%	#	%	#
Rule-based	64.61	84	17.69	23	13.07	17	4.61	6
LLM-based	98.14	106	0.92	1	0	0	0.92	1

Table 3: KEE results. Results are represented in percentages (%) and absolute numbers (#).

all, some, or none of the elements to be identified have been detected, respectively. Finally, it is worth mentioning that, due to out-of-scope causes, in both systems there have been elements that have been erroneously sent as input to the KEE, also presented in Table 3. These interactions, despite having had an output from the KEE, have not been taken into account in this analysis as NLU as they are caused by external errors.

All in all, we can observe a better performance of the LLM-based approach for key element extraction. More specifically, the LLM-based method outperforms the rule-based approach by a 33.5% in terms of fully detected key elements. Furthermore, the rule-based approach is more prone to partially detected and not detected elements, a situation with is rare in the LLM-based approach, with an only case of the former and no cases in the latter, which emphasises the capacity of these methods in this type of tasks.

As for the polarity component, the results have reported a 100% accuracy in both approaches.

5 Conclusions

This work introduces a knowledge graph-based method for managing the knowledge base of a task-oriented dialogue system for industrial settings, in which the knowledge graph is in charge of storing both domain and dialogue-management-related knowledge. This dialogue system features incremental learning capabilities that, by using the HIL/OIL paradigms, allows, on the one hand, for users to give feedback regarding the output of the system and, on the other hand, for experts to improve the knowledge included in the knowledge graph according to operators' feedback. For this, the ontology used in the knowledge graph, which originates from an existing ontology for task-oriented dialogue systems, has been extended to cover the addition of knowledge and the generation of additional dialogues for that end.

Furthermore, for the natural language understanding (NLU) module, which originally was designed by following a rule-based approach, has been implemented by using Large Language Models (LLMs) to improve both the system's maintenance and the quality of the interpretations obtained by it.

The system has been evaluated in two real-world industrial settings: a bin-picking scenario, in which the NLU component was implemented by using LLMs, and a manufacturing scenario, in which the incremental learning capabilities of the system have been tested. For the first scenario, the results show that the performance of LLM-based NLU is higher than the rule-based approach by 16%, which is a significant improvement, especially for the fact that LLMs are easier to adapt to other scenarios than rules. For the second scenario, the addition of feedback interfaces has allowed to improve the existing knowledge graph of the system. The result of this is the addition of more explanations to existing procedures and even two new procedures; all in all, this translates into 25% more knowledge than at the time of the experimentation. This is expected to impact positively in the system's performance from now on, which will be evaluated in a new experimentation task as part of future work.

These results show that the use of knowledge graphs for managing the knowledge base of task-oriented dialogue systems in industrial settings is a promising approach, especially when combined with incremental learning capabilities, and that the use of LLMs for other modules of the system leads to systems that are easy to maintain over time and to adapt to new scenarios.

Acknowledgments

This project has received funding from the Department of Economic Development and Infrastructure of the Basque Government under grant number KK-2022/00102 (BERREKIN), KK-2023/00012 (BEREZ-IA) and KK-2024/00064 (IKUN).

References

János Abonyi, László Nagy, and Tamás Ruppert. 2024. *Knowledge Graph-Based Framework to Support the Human-Centric Approach*, pages 127–156. Springer Nature Switzerland, Cham.

Cristina Aceta, Izaskun Fernández, and Aitor Soroa. 2021. TODO: A Core Ontology for Task-Oriented Dialogue Systems in Industry 4.0. In *Further with Knowledge Graphs*, pages 1–15. IOS Press.

Cristina Aceta, Izaskun Fernández, and Aitor Soroa. 2022. KIDE4I: A generic semantics-based task-oriented dialogue system for human-machine interaction in industry 5.0. *Applied Sciences*, 12(3):1192.

Vinsen Marselino Andreas, Genta Indra Winata, and Ayu Purwarianti. 2022. A comparative study on language models for task-oriented dialogue systems. *CoRR*, arXiv abs/2201.08687.

Lang Cao. 2023. Diaggpt: An llm-based chatbot with automatic topic management for task-oriented dialogue. *arXiv preprint arXiv:2308.08043*.

Songbo Hu, Xiaobin Wang, Zhangdie Yuan, Anna Korhonen, and Ivan Vulić. 2024. DIALIGHT: Lightweight Multilingual Development and Evaluation of Task-Oriented Dialogue Systems with Large Language Models. *arXiv preprint arXiv:2401.02208*.

Vojtěch Hudeček and Ondřej Dušek. 2023. Are large language models all you need for task-oriented dialogue? In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 216–228.

Chen Li, Xiaochun Zhang, Dimitrios Chrysostomou, and Hongji Yang. 2022. Tod4ir: A humanised task-oriented dialogue system for industrial robots. *IEEE Access*, 10:91631–91649.

Bing Liu and Sahisnu Mazumder. 2021. Lifelong and continual learning dialogue systems: learning during conversation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15058–15063.

László Nagy, János Abonyi, and Tamás Ruppert. 2024. Knowledge Graph-Based Framework to Support Human-Centered Collaborative Manufacturing in Industry 5.0. *Applied Sciences*, 14(8).

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Sinan Ozdemir. 2023. *Quick Start Guide to Large Language Models: Strategies and Best Practices for Using ChatGPT and Other LLMs*. Addison-Wesley Professional.

María Poveda-Villalón, Alba Fernández-Izquierdo, and Raúl García-Castro. 2019. Linked Open Terms (LOT) Methodology. *https://doi.org/10.5281/zenodo.2539305*.

Ivan Sekulić, Silvia Terragni, Victor Guimarães, Nghia Khau, Bruna Guedes, Modestas Filipavicius, André Ferreira Manso, and Roland Mathis. 2024. Reliable LLM-based User Simulator for Task-Oriented Dialogue Systems. *arXiv preprint arXiv:2402.13374*.

Mohammed Sidi Yakoub, Sid-Ahmed Selouani, and Roger Nkambou. 2015. Mobile spoken dialogue system using parser dependencies and ontology. *International Journal of Speech Technology*, 18:449–457.

Milene Santos Teixeira, Vinícius Maran, and Mauro Dragoni. 2021. The interplay of a conversational ontology and AI planning for health dialogue management. In *Proceedings of the 36th annual ACM symposium on applied computing*, pages 611–619.

Mathilde Veron, Sahar Ghannay, Anne-Laure Ligozat, and Sophie Rosset. 2021. Lifelong learning and task-oriented dialogue system: what does it mean? In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction: 10th International Workshop on Spoken Dialogue Systems*, pages 347–356. Springer.

A Example procedure

Listing 1: Snippet of the instances in the “Editar o modificar un programa” (“Edit or modify a procedure”) procedure. This example is presented in TTL format for readability.

```
[...]

###  https://w3id.org/todo/tododw-ekin-inst#
    ↪ Method0_Procedure_65113d9f5d9c3075571719b5
:Method0_Procedure_65113d9f5d9c3075571719b5 rdf:type owl:NamedIndividual ,
         tododwHowto:Method ;
     var:hasFirstSegment :Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ;
     var:isMadeOf :Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ;
     var:processSegmentId "0" .

###  https://w3id.org/todo/tododw-ekin-inst#
    ↪ PSI0_Step1_Task0_Method0_Procedure_65113d9f5d9c3075571719b5
:PSI0_Step1_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 rdf:type owl:
    ↪ NamedIndividual ,
         <http://www.mesa.org/xml/B2MML-V0600#ProcessSegmentInformation> ;
     var:relatedToProcessSegment :
         ↪ Step1_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ;
     var:taskImage "https://server/editar-programa/Metodo1-Paso2.png" ;
     tododwHowto:index "0" .

###  https://w3id.org/todo/tododw-ekin-inst#Procedure_65113d9f5d9c3075571719b5
:Procedure_65113d9f5d9c3075571719b5 rdf:type owl:NamedIndividual ,
         tododwHowto:Procedure ;
     var:description "Editar o modificar un programa" ;
     var:hasFirstSegment :Method0_Procedure_65113d9f5d9c3075571719b5 ;
     var:isMadeOf :Method0_Procedure_65113d9f5d9c3075571719b5 ;
     var:processSegmentId "65113d9f5d9c3075571719b5" .

###  https://w3id.org/todo/tododw-ekin-inst#
    ↪ Step0_Task0_Method0_Procedure_65113d9f5d9c3075571719b5
:Step0_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 rdf:type owl:NamedIndividual
    ↪ ,
         tododwHowto:Step ;
     var:description "Abrir el programa deseado. " ;
     var:isPrevious :Step1_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ;
     var:processSegmentId "0" ;
     tododwHowto:hasAssociatedProcedure :Procedure_65113a0c5d9c3075571719b2 .

###  https://w3id.org/todo/tododw-ekin-inst#
    ↪ Step1_Task0_Method0_Procedure_65113d9f5d9c3075571719b5
:Step1_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 rdf:type owl:NamedIndividual
    ↪  ,
         tododwHowto:Step ;
     var:description "Con el programa en pantalla, tal y como se muestra en la
         ↪ siguiente imagen, se podrá comenzar a modificar o extender el código G
         ↪ para programar la pieza." ;
     var:hasRelatedInformation :
         ↪ PSI0_Step1_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ;
     var:isNext :Step0_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ;
     var:isPrevious :Step2_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ;
     var:processSegmentId "1" .


###  https://w3id.org/todo/tododw-ekin-inst#
    ↪ Step2_Task0_Method0_Procedure_65113d9f5d9c3075571719b5
:Step2_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 rdf:type owl:NamedIndividual
    ↪  ,
         tododwHowto:Step ;
     var:description "Para guardar los cambios, no es necesaria ninguna acción especí
         ↪ fica: se guarda automáticamente." ;
     var:isNext :Step1_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ;
     var:processSegmentId "2" .
```

```

###   https://w3id.org/todo/tododw-ekin-inst#
    ↪ Task0_Method0_Procedure_65113d9f5d9c3075571719b5
:Task0_Method0_Procedure_65113d9f5d9c3075571719b5 rdf:type owl:NamedIndividual ,
        tododwHowto:Task ;
    var:hasFirstSegment :Step0_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ;
    var:isMadeOf :Step0_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ,
                 :Step1_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ,
                 :Step2_Task0_Method0_Procedure_65113d9f5d9c3075571719b5 ;
    var:processSegmentId "0" .

[...]
```