Knowledge Retrieval with LLMs using Context-Specific Intent and Slot Classification

Julian Schubert[1], Markus Krug[1] and Joachim Baumeister[1,2]

[1]denkbares GmbH, John-Skilton-Straße 8, 97074 Würzburg
[2]University of Würzburg, Am Hubland, 97076 Würzburg

Abstract
A significant proportion of applications in the industrial domain rely on RAG and synthesise structured knowledge into strings of text. This approach, however, occasionally produces answers that are factually incorrect. This limits the applicability of LLMs in industrial settings, where factual correct answers are a critical requirement. To address this issue, we propose a system for intent-based closed question answering that can be applied when structured factual knowledge is available, for instance in a Knowledge Graph. This system first employs an intent grammar to narrow down the possible user intents. The slots of each intent can then be filled context-sensitively, with the aim of improving upon existing NER techniques. Following a verification step during which the model is prompted to ensure that the correct intent and entities have been extracted, a SPARQL query can be executed and verbalised using rule-based methods.

Keywords
Large Language Models, Context-Specific Intent and Slot Classification, CEUR-WS

1. Introduction

The recent success of Large Language Models (LLMs) revitalized the progress in many fields in Natural Language Processing. As LLMs predominantly work with textual outputs it is almost natural to utilize them as Chatbots [1, 2]. Albeit their impressive capabilities of forming coherent and logical texts, LLMs struggle with factuality [3] and solving complex tasks that require planning and reasoning [4, 5, 6]. Chatbots in the industrial domain can revolutionize the way internal data is accessed - it is natural to us to satisfy our information needs using natural language, the main interface of an LLM. As factual incorrect answers pose severe risks in industrial settings, most current applications of LLMs in the industry utilize an approach based on Retrieval Augmented Generation (RAG) [7, 8]. Standard RAG incorporates knowledge stored in private data into the reasoning process of a LLM. One rather paradox consequence of RAG is that structural data, such as data bases and knowledge graphs, has to be converted into raw text in order for the LLM to understand and make use of the data. The risk of non-factual information is mitigated by also presenting the data the information is obtained from to the user. As this concept is still rather new in industrial settings such AI based systems have yet to earn their trust by the users. In this positional paper we outline an approach that allows

LWDA'24: Lernen, Wissen, Daten, Analysen. September 23–25, 2024, Würzburg, Germany
julian.schubert@denkbares.com (J. Schubert); markus.krug@denkbares.com (M. Krug); joachim.baumeister@denkbares.com (J. Baumeister)

the combination of structural data, in our case represented as knowledge graphs, and LLMs in the setting of Information Retrieval (more precisely question answering). Our proposed approach mitigates two of the aforementioned main weak spots of LLMs: (i) Non-factuality, and (ii) inability to plan.

2. Related Work

This section provides a brief overview of RAG, along with an examination of recent advancements in intent-based closed question answering.

2.1. Retrieval-Augmented Generation

In contrast to fine-tuning an LLM for a specific domain, RAG is a method that aims to improve model answers by enriching the context of the LLM with useful information. RAG works through an initial retrieval step where an external data source is queried to gather and rank relevant information. This information is then used to inform the subsequent generation phase. The incorporation of retrieved evidence into the responses significantly enhances the accuracy and relevance of the generated answer [7, 9].

2.2. Intent-Based Closed Question Answering

Rather than relying on the model to directly generate an accurate response from the provided documents, intent-based question answering employs a model to narrow down the user's intent and then attempts to resolve this given intent. For instance, Chen et al. fine-tuned a BERT-Model [11] to jointly model intent classification and slot filling on the ATIS Dataset [12]. To enhance the modelling of the relationships between slots and intents, Qin et al. proposed the utilisation of an LSTM [14] to construct an adaptive intent-slot graph interaction layer. This layer explicitly establishes correlations between slot and intent nodes. Furthermore, graph attention networks are applied to each token, enabling the model to capture and integrate fine-grained intent information for precise token-level slot prediction.

2.2.1. Using Named Entity Recognition for Slot Filling

Each intent has various slots that need to be filled in order to answer the input query. For example, the intent `Repair Component` has a slot with the identifier of the component that should be repaired. We propose using Named Endtity Recognition (NER) to fill these slots and will therefore give a briev overview of recent developmends in this field. GPT-NER transforms the traditional sequence-labeling task of NER into a generation task. Instead of making a prediction for each token, GPT-NER employs special markings within the generated text to identify named entities. Following this prediction step, a verification step is performed where the model is prompted to confirm the accuracy of the extracted entities. [15, 16]. Our proposed approach builds upon the success of applying LLMs for NER. In contrast to general NER, our system is limited to a certain domain allowing for context-specific prompting and therefore, potentially further improvement of existing results.

3. Approach

Generating factual correct outputs from an LLM is very hard in an open setting as the amount of potential queries is unrestricted. Therefore, we restrict queries to a pre-defined set of user intents and use RAG as a fallback for queries outside of the current scope of our system. In this positional paper we will only focus on outlining a model for the technical service in the engineering domain that requires structured data to be available, for instance in a Knowledge Graph (KG). Our proposed system starts by narrowing down the possible user intents and then attempts to resolve the slots for each of the intents, verifying solutions by re-prompting the LLM in a context-sensitive manner. This explicitly means that we avoid using the LLM to plan but rather provide a plan in the form of a domain-specific grammar to identify the correct intent. To avoid using a LLM to generate the final output, which might again introduce hallucinations or incorrect facts, we propose to link a SPARQL-Query [17] to each intent that can be executed once all slots are filled. The result of this query can then be presented in a structured manner (e.g. in a table), ensuring factual correct outputs of the proposed system.

3.1. Modeling Technical Service in the Engineering Domain

We elaborate our approach by providing a small example model of the technical service in the engineering domain, comprising all relevant concepts: `components`, `function`, `facts` and `activities` as shown in figure 1. This domain is of particular interest as the main challenge in the technical service is to quickly provided factual correct support to the customer in order to resolve technical problems with a product. We illustrate the introduced concepts by a simple example of a lamp and the technical service of this lamp, respectively.

3.1.1. Component

First, we define a `component` as a distinct, identifiable part or element of a larger system or machine. Additionally, a component often consists of various sub-components (referred to as child-nodes) and is part of a larger assembly or system (referred to as the parent-node). For example, the base of a lamp could be a component, with the lamp itself being the parent-component of the base. The wireless charger and the weight are two child components of the base.

3.1.2. Functions

A `function` of a component describes the intended role or purpose of the component within the larger system or machine. In the case of a lamp, for instance, this could be to provide light for the room or to charge the phone.

3.1.3. Facts

`Facts` are specific pieces of information about the component. These details provide essential data that describe various attributes and characteristics of the component. Examples of facts

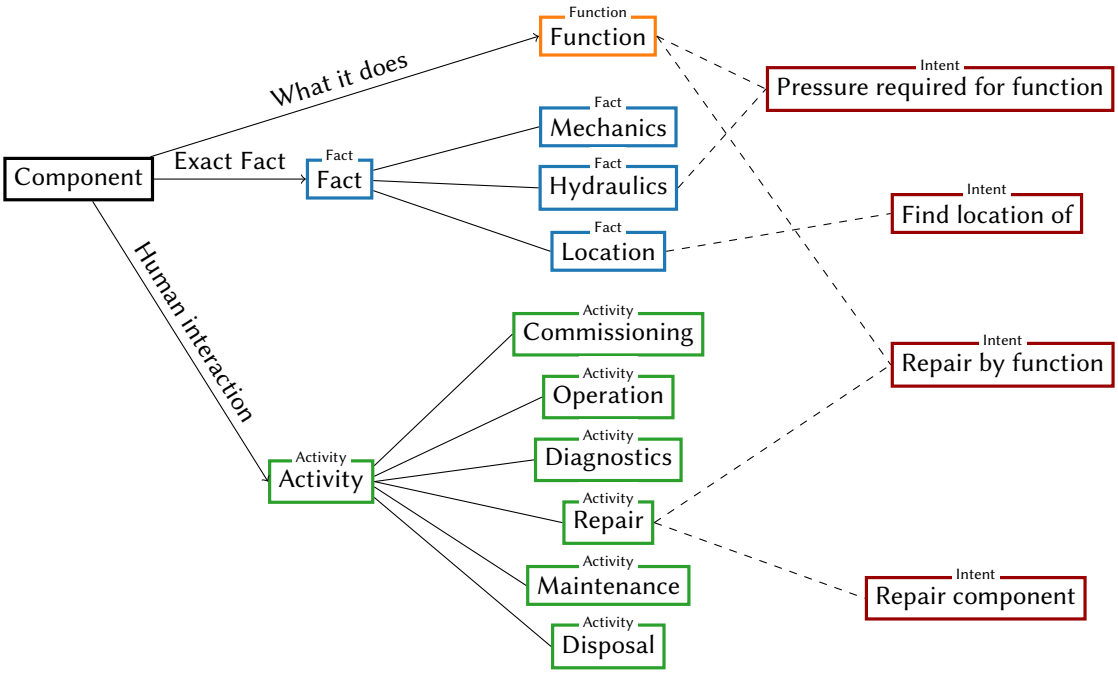

Figure 1: Model of the technical service in the engineering domain. `Components` have `functions`, `facts` and `activities` attached to them. Each possible `user-intent` is connected to the relevant conepts.

include the weight of the component, the brightness of the bulb or output power of the wireless charging pad.

3.1.4. Activities

An `activity` encompasses all the actions a user can perform with the component and are grouped in six different categories:

Commissioning covers everything that must be done before the component is used for the first time. It includes initial setup, calibration, and any preliminary checks required to ensure the component is ready for operation. For a lamp, this might involve the assembly of the component and its connection to the power source.

Operation describes detailed instructions on the proper usage techniques, operating conditions, and any specific procedures to follow to ensure efficient and safe operation. Typical operation activities include turning the lamp on and adjusting the brightness.

Diagnostics involves identifying and troubleshooting faults within the component, including methods for detecting issues, tools required for diagnostics, and procedures to follow for accurate fault identification. For instance, checking if the lamp fails to turn on due to a blown fuse or a faulty bulb.

Repair outlines the steps necessary to repair a defective component. It includes detailed repair

procedures, tools, and parts required. One common repair activity is changing the bulb of a lamp.

Maintenance focuses on the preventive measures needed to keep the component in good working condition. It includes regular maintenance tasks, schedules, and instructions on how to perform upkeep activities to prevent breakdowns and extend the component's lifespan. Regularly cleaning the lamp and ensuring the connections are secure are key maintenance tasks.

Disposal describes how to properly dispose of the component when it is no longer usable. It includes guidelines for environmentally friendly disposal, regulatory compliance, and any special handling required for hazardous materials. Disposing of the lamp at a recycling center that handles electronic waste is an example of proper disposal.

In addition to these categories, several important aspects enhance the definition and execution of activities. An activity can have a warning attached, alerting users to potential hazards or safety precautions that need to be taken. Activities may also have conditions attached, indicating that certain activities can only be performed under specific circumstances or environmental conditions.

Typically, an activity consists of multiple steps that must be performed in a specific order to ensure proper execution and achieve the desired outcome. Furthermore, some activities may optionally require additional parts or tools to be completed successfully.

3.2. User Intents for Technical Service

Based on this abstract view of the technical service domain, it is possible to define and categorise various different `user intents`, where one intent can possibly consist of multiple sub-intents. For instance, a user may want to know which component has a given function and subsequently how this component can be repaired. In this scenario, `Repair by function` could be considered the primary intent, with `Repair component` acting as a potential sub-intent. This hierarchical structure of intents and sub-intents facilitates the organisation and effective addressing of complex queries, ensuring that each aspect of the user's question is comprehensively answered.

Each intent may have one or more slots, which are entities or aggregations of entities that are necessary to answer the query and provide essential context for generating accurate and relevant responses. For instance, when a user inquires about how to repair a specific component, one slot could be identifying the component in question.

3.2.1. Finding the correct User Intents

In our proposed model, each concept is associated with multiple intents, with the potential for one intent to be assigned to multiple concepts simultaneously, reflecting the multifaceted nature of user inquiries in the technical service. For example, as shown in figure 1, the intent `Find location of` is only assigned to the `Location`-Concept. In contrast, the intent `Repair by function`, was assigned to both `Repair` and `Function`. For a given user input, our proposed algorithm initially narrows down all possible intents by traversing down the tree. It is hereby important to note that an intent can be found via various branches, reducing

the risk of excluding the correct intent due to one classification error. Subsequently, for each remaining intent, the algorithm attempts to fill the slots using the provided context (see §3.2.2). In the event that no intent has all slots filled, the standard RAG approach is employed as a fallback. When multiple intents have been successfully filled, the LLM can be prompted again to ascertain which of the intents is correct, or the user can be prompted to select the desired intent.

3.2.2. Filling Intent Slots

The approach leverages the intent to perform context-sensitive NER. By incorporating context from the intent into the prompt, the NER is tailored to the specific context. For instance, given the intent `Repair by Function`, the model can be prompted to extract only the exact function the user desires to repair, rather than extracting all functions mentioned within the text. As the entities extracted by the LLM might not match the graph labels, we propose generating embeddings for graph entities and for the model output. By comparing these embeddings, the closest matches can be found and used to re-prompt the model to output the correct label as specified in the graph. This process avoids adding numerous concepts to the model's context and ensures alignment with the graph database.

4. Promises

The approach detailed in this work promises advancements for both - academics and industry. In the academic context, the following research questions are of particular interest:

- Comparing context-specific NER with traditional NER based on Large Language Models (LLMs).
- Evaluating a domain-specific approach for integrating structured knowledge into RAG architectures.
- Extending the traditional NER approach to include a dialog mechanism aimed at satisfying the user's information needs.
- Comparing methodologies for ensuring the security of the output information.

For the industry, a framework can be developed that allows the integration of potentially existing structured information within the company into the context of RAG, without relinquishing complete control over its correctness. This framework can be easily extended to other domains, as incorporating a new domain would simply require adding the necessary intents. Beyond question answering, the framework's intent classification could also be used to classify documents of any kind, improving the retrieval step in RAG by pre-classifying documents by intent, classifying the query, and adding only documents with matching intents to the context. This enables applications to reliably satisfy information needs expressed in natural language.

References

[1] J. Wei, S. Kim, H. Jung, Y.-H. Kim, Leveraging large language models to power chatbots for collecting user self-reported data, Proceedings of the ACM on Human-Computer Interaction 8 (2024) 1–35. URL: http://dx.doi.org/10.1145/3637364. doi:10.1145/3637364.

[2] G. Lee, V. Hartmann, J. Park, D. Papailiopoulos, K. Lee, Prompted llms as chatbot modules for long open-domain conversation, in: Findings of the Association for Computational Linguistics: ACL 2023, Association for Computational Linguistics, 2023. URL: http://dx.doi.org/10.18653/v1/2023.findings-acl.277. doi:10.18653/v1/2023.findings-acl.277.

[3] Y. Wang, M. Wang, M. Arslan Manzoor, F. Liu, G. Georgiev, R. Jyoti Das, P. Nakov, Factuality of Large Language Models in the Year 2024, arXiv e-prints (2024) arXiv:2402.02420. doi:10.48550/arXiv.2402.02420. arXiv:2402.02420.

[4] S. Kambhampati, K. Valmeekam, L. Guan, K. Stechly, M. Verma, S. Bhambri, L. Saldyt, A. Murthy, Llms can't plan, but can help planning in llm-modulo frameworks, arXiv preprint arXiv:2402.01817 (2024).

[5] S. Kambhampati, Can large language models reason and plan?, Annals of the New York Academy of Sciences 1534 (2024) 15–18. URL: https://nyaspubs.onlinelibrary.wiley.com/doi/abs/10.1111/nyas.15125. doi:https://doi.org/10.1111/nyas.15125.

[6] F. Chollet, On the measure of intelligence, 2019. URL: https://arxiv.org/abs/1911.01547. arXiv:1911.01547.

[7] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in Neural Information Processing Systems 33 (2020) 9459–9474.

[8] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, Q. Guo, M. Wang, H. Wang, Retrieval-augmented generation for large language models: A survey, 2024. arXiv:2312.10997.

[9] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, Retrieval-augmented generation for large language models: A survey, arXiv preprint arXiv:2312.10997 (2023).

[10] Q. Chen, Z. Zhuo, W. Wang, Bert for joint intent classification and slot filling, 2019. URL: https://arxiv.org/abs/1902.10909. arXiv:1902.10909.

[11] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: J. Burstein, C. Doran, T. Solorio (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. URL: https://aclanthology.org/N19-1423. doi:10.18653/v1/N19-1423.

[12] C. T. Hemphill, J. J. Godfrey, G. R. Doddington, The ATIS spoken language systems pilot corpus, in: Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990, 1990. URL: https://aclanthology.org/H90-1021.

[13] L. Qin, X. Xu, W. Che, T. Liu, Agif: An adaptive graph-interactive framework for joint multiple intent detection and slot filling, 2020. URL: https://arxiv.org/abs/2004.10087. arXiv:2004.10087.

[14] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 1735–1780. doi:10.1162/neco.1997.9.8.1735.

[15] S. Wang, X. Sun, X. Li, R. Ouyang, F. Wu, T. Zhang, J. Li, G. Wang, Gpt-ner: Named entity recognition via large language models, 2023. URL: https://arxiv.org/abs/2304.10428. arXiv:2304.10428.

[16] M. Li, R. Zhang, How far is language model from 100% few-shot named entity recognition in medical domain, 2024. URL: https://arxiv.org/abs/2307.00186. arXiv:2307.00186.

[17] W3C SPARQL Working Group, Sparql 1.1 query language, https://www.w3.org/TR/sparql11-query/, 2013. Accessed: 2024-07-08.

A. Preliminary Case Study: denkLampe - Graph

In this chapter we will introduce the **denkLampe**, a lamp that was modeled in the previously defined scheme. After that, we will provide a minimal grammar to resolve intents on this domain and show one sample trail.

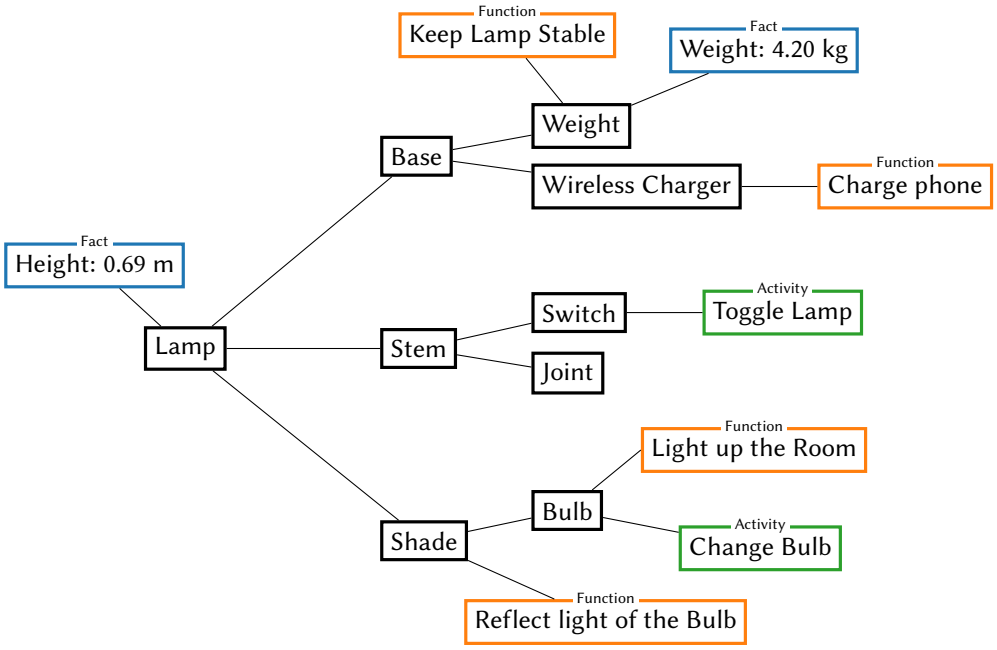

Figure 2: Simplified Lamp-Model. The lamp consists of multiple `components`: the base, stem and a shade which each have sub-components attached to. `Facts`, like the height of the lamp, are colored blue, `functions`, like charging the phone, are colored orange and `activities` like replacing the bulb are colored green.

A.1. DenkLampe

As shown in figure 2, the denkLampe consists of a base, stem and shade. In this example, the base of the lamp consists of a weight which keeps the lamp stable as well as a wireless charger. The stem of the lamp has a switch to toggle the lamp as well as the joint as sub-components. Lastly, the bulb of the lamp is the only child-node of the shade and has the function of lighting up the room.

A.2. Grammar

In the first step, our grammar decides if the user question revolves around a function, fact or activity. This could for example be done by prompting a LLM to decide for each concept, if it is relevant for the given user query or not. One possible way to do this could be to ask the model simple yes-no questions, demonstrated in figure 3. After executing a similar query for the facts

Prompt

You are denkender, a helpful assistant that helps determining the intent of a user. It is very important that you do not answer the question of the user directly, but instead you determine if the user wants to know something related to the function of a component. A function of a component is always something the component does.
For example:

> If the user asks "How heavy is the lamp?", your answer should be "No"
> If the user asks "What component lights up the room?", your answer should be "Yes"

Remember to only answer with "Yes" or "No" instead of answering the question of the user.

Figure 3: Sample prompt for a LLM to determine if the concept function is relevant for a given query.

and activities, the algorithm now explores all relevant branches of the tree. After that, we can further narrow down the exact category of fact or activity using similar prompting. Once we are at a leaf-node, we can prompt the model for each remaining intent, if this is the correct intent for the user query. For each intent selected by this procedure, we attempt to fill out the slots as described below. The key-change to existing work in this field is that we do not use the model for planning but only to select the correct path according to a pre-defined grammar.

A.3. Filling Slots

Given an intent selected to be resolved by our grammar, our system then attempts to fill all slots for this intent. For example, for the user query `How do I change the thing that lights up the room`, only the slot `Function of the component that needs to be repaired` needs to be filled, given that our system correctly identified the intent `Repair by function`. A possible query to fill this slot is shown below:

A.4. Demo Trail

Lastly, we will provide a short demo-trail on how our proposed approach could resolve the user query `The lamp does not light up, how can I fix that?`. First, our system needs to decide if a function, fact and activity are relevant. In this case, the function is relevant as the user is asking for the function `Lighting up the room`. There is no fact relevant for this function. However, an activity is relevant for this query as the user wants to do something with the lamp, he wants to repair it which is an activity. The result of this first step is illustrated in figure 5. Next, the algorithm has to narrow down the specific kind of activity. In this case, the user wants to repair something, so `Repair` is the only remaining activity, shown in figure 6. In this case, only the two intents `Repair by function` and `Repair component` remain. As the algorithm already determined that a function is relevant for the given user intent, it can discard the intent `Repair component` as it is not attached to a function. Alternatively, the algorithm could prompt a LLM to determine which of the remaining intents, of which all slots

Prompt

You are denkender, a helpful assistant that helps performing NER for user input. It is very important that you do not answer the question of the user directly, but instead you extract the function of the component the user wants to repair.
For example:

If the user asks "I want to repair the thing that reflects the light of the bulb",
your answer should be "Reflect light of the bulb"
If the user asks "How do i replace the component responsible for charging my phone",
your answer should be "Charge phone"

Remember to only answer with the name of the function of the component the user wants to repair.

Figure 4: Sample prompt for a LLM to extract the relevant function from a given query.

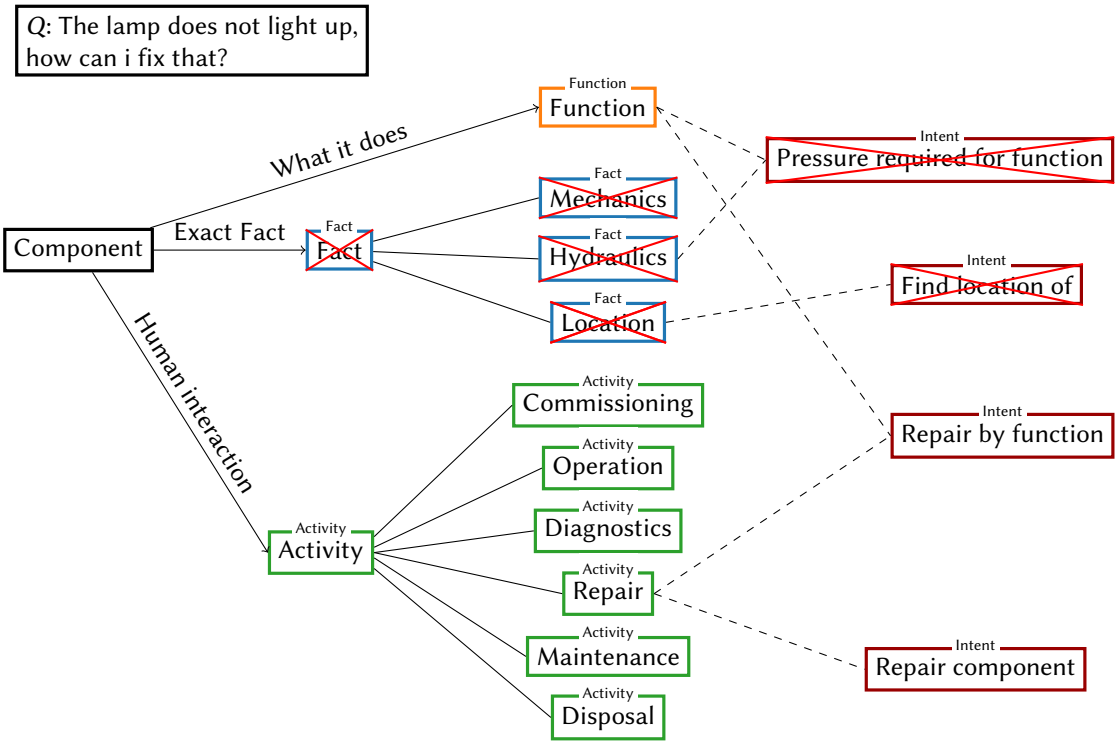

Figure 5: Demo-Trail Step 1: Only the concepts `function` and `activity` are relevant for the given user query.

can be filled, is correct. After filling all slots as shown in §A.3, the answer of the SPARQL-Query attached to the intent can be presented to the user.

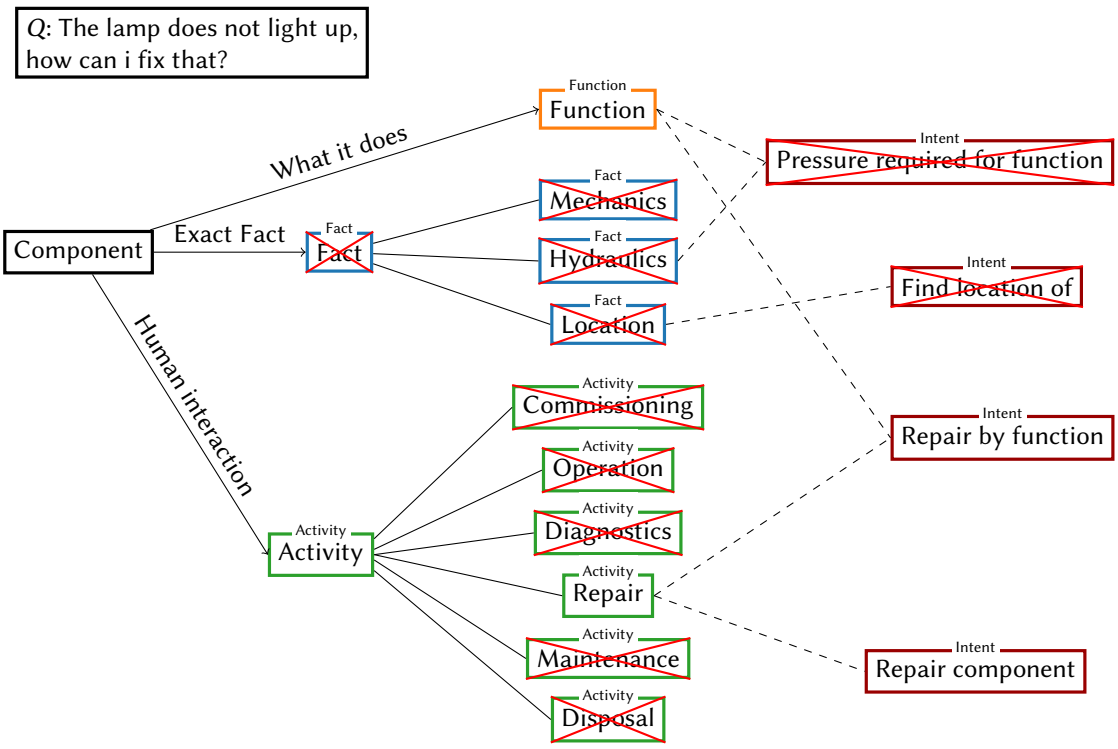

Figure 6: Demo-Trail Step 2: `Repair` is the only relevant activity. Only two possible intents remain.