OntoChat: a Framework for Conversational Ontology Engineering using Language Models

Bohui Zhang[1][0000−0001−5430−1624], Valentina Anita
Carriero[2][0000−0003−1427−3723], Katrin Schreiberhuber[3][0000−0003−1815−8167],
Stefani Tsaneva[3][0000−0002−0895−6379], Lucía Sánchez
González[4][0000−0002−1685−4700], Jongmo Kim[1][0000−0002−4984−1674], and Jacopo
de Berardinis[1][0000−0001−6770−1969]

[1] Department of Informatics, King's College London, London, UK
[2] Cefriel – Politecnico di Milano, Milan, Italy
[3] Vienna University of Economics and Business, Vienna, Austria
[4] Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
`bohui.zhang@kcl.ac.uk`

Abstract. Ontology engineering (OE) in large projects poses a number of challenges arising from the heterogeneous backgrounds of the various stakeholders, domain experts, and their complex interactions with ontology designers. This multi-party interaction often creates systematic ambiguities and biases from the elicitation of ontology requirements, which directly affect the design, evaluation and may jeopardise the target reuse. Meanwhile, current OE methodologies strongly rely on manual activities (e.g., interviews, discussion pages). After collecting evidence on the most crucial OE activities, we introduce **OntoChat**, a framework for conversational ontology engineering that supports requirement elicitation, analysis, and testing. By interacting with a conversational agent, users can steer the creation of user stories and the extraction of competency questions, while receiving computational support to analyse the overall requirements and test early versions of the resulting ontologies. We evaluate OntoChat by replicating the engineering of the Music Meta Ontology, and collecting preliminary metrics on the effectiveness of each component from users. We release all code at `https://github.com/King-s-Knowledge-Graph-Lab/OntoChat`.

Keywords: Ontology Engineering · Large Language Models · Competency Questions · Computational Creativity.

1 Introduction

Ontology Engineering (OE) encompasses a number of activities defining a collaborative effort to design, evaluate, and reuse ontologies of general or domain-specific purposes [19, 25]. Depending on the intended reuse by one or more stakeholders, the process starts with ontology designers eliciting requirements from the former, while interacting with experts to validate their formal understanding of the domain. Although the way OE activities are organised may vary depending

on the methodology adopted [9, 26], these interactions are frequent and iterative throughout the development cycle. In large projects, ontology designers engage in a number of manual activities and interact with multiple parties to make sure requirements are collected consistently and comprehensively, seeking clarification from domain experts to formalise and test them [11]. This multi-party interaction may require substantial resources and create systematic ambiguities and biases arising from potentially conflicting or ill-formulated requirements. Not only does this directly affect the ontology design and evaluation, but it may also jeopardise their reuse by the same stakeholders. In other cases, a blurry line exists among stakeholders, domain experts, and ontology designers, which is common to multidisciplinary and highly collaborative projects, such as Wikidata [29].

OE has long faced arguments about its technical challenges and costs. Despite the development of collaborative methodologies (c.f. Section 2), the field still faces concerns about positively impacting the liveliness, evolution, and reusability of ontologies [20]. Meanwhile, Large Language Models (LLMs) have received increasing attention in the Semantic Web due to their language understanding capabilities. These provide generalisation across various tasks (e.g., question answering, text summarisation), making it possible to effectively generate, process, and annotate text to address knowledge engineering tasks [3, 30].

Here, we hypothesise that LLMs can assist and facilitate OE activities to implement conversational workflows and prompt-driven functionalities that can reduce the complex interactions between domain experts and ontology designers, while accelerating the analysis and formalisation of requirements. To confirm our motivations and collect insights for the design of such a framework, we put forth the following research questions.

- **RQ1**: Which ontology engineering activities are the most in need of computational support?
- **RQ2**: How can LLMs enable a conversational ontology engineering framework to support these activities?

To address RQ1, we conducted a survey asking ontology engineers to rate OE activities in relation to their difficulty and the manual effort required. We reuse the insights collected from the survey to implement a conversational workflow where different categories of users interact independently with the system and produce intermediate artefacts and documentation throughout the various stages of the OE process (RQ2). In sum, this work contributed:

- Evidence on which OE activities are most in need of computational support, gathered from a survey involving participants with OE expertise.
- **OntoChat**, a conversational framework for ontology engineering providing support for (i) *requirement elicitation* (user story creation, competency question extraction), (ii) *analysis* (competency question verification, reduction, and clustering), and for (iii) *testing* preliminary versions of an ontology.
- A preliminary evaluation of an online prototype of OntoChat where participants were asked to replicate the OE of the Music Meta ontology [6] and measure the effectiveness of their outcomes and interactions.

2 Related work

Various OE methodologies have been proposed over the years, with their focus shifting towards collaborative approaches [25]. Early works include METHON-TOLOGY [14], based on requirements elicitation from [27] and providing support for conceptualisation, implementation, and maintenance steps; and DILIGENT [23] which also account for the involvement of different stakeholders. More recent work incorporated Agile principles to support iterative ontology development, such as NEON [26], SAMOD [22], and *eXtreme Design* (XD) [9, 8]. The latter also provides guidelines for requirement elicitation and is strongly test-based.

In [5], ontology requirements are collected from *customers* in the form of *user stories*. A story[5] contains three main components: the *persona* portrays a typical user, including their name, occupation, skills, and interests; the *goal* captures the persona's aims in the story; the *scenario* describes how the persona's goals are currently addressed, to contextualise the gap with the resource being developed. Through the collection of stories, ontology requirements can then be defined by extracting competency questions (CQs) – the natural language counterpart of structured queries that the resulting knowledge graph (KG) should answer [16].

CQs are central to OE. They facilitate the collection of requirements, drive the implementation of the ontologies (e.g., in XD, they are mapped to ontology design patterns) and are used for testing [10]. Nonetheless, despite the level of experience in a particular OE methodology, bottlenecks, ambiguities, and domain jargon can still hamper progress from the requirement collection stage [11]. This especially happens when several stakeholders and domain experts are involved. To mitigate this issue, [5] introduced IDEA – a tool supporting the iterative elicitation and improvement of CQs via NLP methods.

Recently, LLMs have been applied to support and augment knowledge engineering tasks. Applications range from representing domain knowledge and generating examples of classes and relations to providing explanations and recommendations on ontologies after verbalising them into plain text [21]. For ontology engineering, LLM-based approaches have shown promising results for ontology matching and alignment [24, 17, 18]. Other studies have focused on ontology construction and learning from text, using LLMs to suggest relevant subconcepts [15] and to automatically extract and structuring knowledge [4]. In the context of requirement elicitation, new methods were contributed for retrofitting CQs from ontologies to promote reuse [2] and extracting CQs directly from KGs [13]. Overall, these works demonstrated great potential for LLM-driven knowledge engineering, but also acknowledged significant issues such as hallucination, poor non-linguistic reasoning, and the high cost of fine-tuning.

3 OntoChat: towards conversational ontology engineering

To address RQ1 and inform the design of our framework, we conducted an online survey asking participants to rank OE activities for complexity and need of

---

[5] Examples of user stories at `https://github.com/polifonia-project/stories`
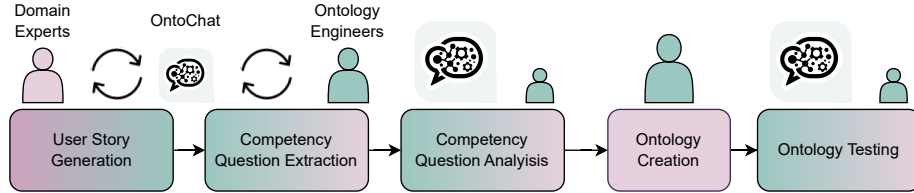
Fig. 1: Illustration of the workflow alongside the main features in OntoChat.

computational support. The survey was conducted via Google Forms (without recording any personal data from participants) and was distributed to Semantic Web practitioners. Participants were asked to quantify the agreement of statements on a 5-point Likert scale, with each statement expressing an OE activity.

We gathered responses from $N = 23$ participants with various level of experience and familiarity with OE methodologies (primarily Neon, Ontology 101, Linked Open Terms). Results are detailed in Appx A. Given the size of our sample, we decided to rely on strong evidence ($\geq 75\%$ positive responses using 4-5 scores), finding that the most demanding OE tasks in need of computational support are: the collection of ontology requirements (86.4%), the extraction of CQs from textual ontology requirements (81.8%), the analysis of ontology requirements (77.3%), and ontology testing (81.8%).

Based on these findings, and building upon the IDEA approach [5], we designed OntoChat to support such OE activities. The workflow, illustrated in Fig. 1, leverages LLMs as knowledge elicitators to reduce the demand and complexity of the multi-party interactions with ontology designers, and aims at accelerating OE tasks. To collect requirements, the process starts with stakeholders and domain experts co-creating user stories by interacting with our conversational agent. Ontology engineers can then extract CQs by iteratively refining the model's recommendations; reduce redundant requirements, and analyse the resulting CQs via clustering. Finally, OntoChat also allows testing preliminary versions of ontologies via verbalisation and unit prompting.

3.1 Assisted persona and story creation

To enable the collaborative user story generation OntoChat follows steps below.

Step 1: LLM role definition (back-end). The LLM behaviour is primed to emulate a knowledge elicitator aiming to gather information from the user about each user story component – the persona definition, specification of concrete goals & addressed scenario, and the provision of data examples (back-end).

Step 2: Knowledge elicitation (user involvement). The user is guided through a series of questions focusing on a particular story aspect. For instance, to gather insights about the persona's background, OntoChat asks *"What are the name, occupations, skills, interests of the user?"* Whenever the user does not provide enough details or gives partial answers, the LLM continues to elicit additional insights until all necessary information is gathered (c.f. Fig. 7b, Appx. C.1).

Step 3: User story generation (back-end) The initial user story draft is created following a one-shot learning approach [12]. The LLM is supplied with a user story example, and is prompted to follow the same structure for generating a *user story draft* summarising the information extracted in the elicitation step.

Step 4: Refinement (user involvement). The user is presented with the user story draft and is encouraged to provide feedback. The refinement stage is iterative and continues until the user no longer requests further changes. Possible refinements include the correction of factual inconsistencies, additions to the user stories, removal of irrelevant details, etc. We provide some exemplary refinement requests in Fig. 8 (Appx. C.1).

3.2 Competency question extraction

The main objective of this module is to assist ontology engineers in extracting CQs from user stories. The procedure is organised as follows.

Step 1: Instructing the LLM for CQ extraction (back-end). The model is provided with examples of pairs of user story fragments and expected CQs, to align its outputs to the expectations of ontology engineers.

Step 2: First extraction of CQs (user involvement). The ontology engineer is asked to provide a user story for CQ extraction. The user story may be manually crafted, or obtained from the previous step (see Fig. 9, Appx. C.1). As output, OntoChat provides an initial list of CQs (see Fig. 10, Appx. C.2)

Step 3: Competency question refinement (back-end). The LLM is provided with a series of prompts (hidden from the user) to perform two *refinement steps*.

– *Step 3.1: Split not atomic CQs.* If the example data is complex, users may get non-atomic questions from a single example. As these usually entail nested requirements, complex CQs need to be split. Following a few-shot learning approach, the LLM is asked whether each CQ has a complex form, hence triggering the simplification. For example, from the data "*The musical work Penny Lane has genre/style baroque pop and psychedelic pop.*", the LLM generated "*What genres/styles are associated with Penny Lane?*". After this step, the LLM replaces it with two distinct CQs: "*What genres are associated with Penny Lane?*" and "*What styles are associated with Penny Lane?*".
– *Step 3.2: Named entities abstraction.* As an example, the previous CQs replaced the specific genres with the interrogative pronoun "what" (genres/styles). However, it did not remove the real-world entity "Penny Lane". Within this step, the LLMs is prompted to check again the CQs, and, guided by examples, remove possible named entities. This yields abstract CQs like "*What genres are associated with the musical work?*" and "*What styles are associated with the musical work?*".

Step 4: User confirmation (user involvement). Finally, OntoChat asks the user whether the number of CQs and their formulation are sound. If not, by leveraging knowledge acquired from previous prompts (see Step 3), the model repeats the refinement steps until the user is satisfied (see Fig. 11, Appx. C.2).

3.3   Competency question filtration and analysis

As some CQs may be redundant or show negligible semantic variations that are of little interest to ontology engineers, OntoChat provides support for their filtration and analysis. This is achieved through: *paraphrase identification*, to remove equivalent CQs; and *CQ clustering*, to identify groups of similar requirements. In [13], the former was found to have two benefits: (i) it mitigates the noise and the artefacts introduced in the previous steps; (ii) it reduces the number of CQs that will be presented to ontology engineers.

In contrast to [13, 5], which both rely on sentence embeddings and specialised models, this functionality is entirely supported by LLMs, which is motivated by recent findings demonstrating that LLMs possess clustering capabilities [1, 31, 28]. Given a list of CQs, the LLM is asked to remove redundant questions and find meaningful groups of CQs sharing the same thematic focus and intent. The latter is expected to support ontology designers in understanding requirements and possibly organising their Agile teams (e.g., a team receiving a CQ cluster based on their familiarity with the sub-domain). In the current version, this step does not require user supervision.

3.4   Ontology testing support

While the previous functionalities focus on requirement elicitation and analysis, this component provides support for testing preliminary or iterative versions of an ontology. Ontology testing efforts are often categorised into three methodologies: CQ verification, inference verification, and error provocation. The first two are concerned with verifying the correct implementation of a requirement, whereas the latter is needed to find cases where the ontology should fail [10]. These are typically done by formalising CQs into SPARQL queries.

To test preliminary versions of an ontology, we aim for a SPARQL-free approach to achieve fast CQ verification and inference, while supporting error provocation. This is achieved in two steps: *ontology verbalisation*, and prompt-driven *CQ unit testing*. Our verbalisation converts a OWL ontology into plain text by documenting classes, properties, named entities, and their relationships in a descriptional manner. The method follows a simple algorithmic procedure and assumes that the ontology is well commented to produce an expressive verbalisation. Then, using the verbalisation, the LLM is prompted to asses the coverage of each CQ by replying *Yes/No*. To prevent prompt leakage and ensure independence in the model's predictions, this is done separately for each CQ.

3.5   Implementation details

OntoChat is implemented in `Python 3.11` and is released on GitHub [6] under the MIT license. To facilitate its use and collect user feedback, we implemented an interface prototype using `Gradio`[7]. This can be launched on a local server,

---

[6] `https://github.com/King-s-Knowledge-Graph-Lab/OntoChat`
[7] `https://www.gradio.app`

also hosted on Hugging Face Spaces[8]. The interface has four tabs that wrap all the aforementioned functionalities within the same environment. The current version uses OpenAI's API and has been evaluated on the GPT-3 family of models (`gpt-3.5-turbo`, and `gpt-3.5-turbo-16k` for larger contexts).

4   Evaluation

We performed a component-based evaluation of OntoChat to measure the effectiveness of each functionality and collect user feedback based on their experiences. The evaluation was organised to replicate the OE activities of the Music Meta ontology [6]. It was chosen as a benchmark/testbed for three reasons: it required considerable OE efforts and was already the source of ambiguities in the Polifonia project [7]; it was complemented by high-quality material (user stories, CQs, documentation, queries, etc.) from [5] to use as ground truth; the authors had access to a pool of domain experts for evaluation.

4.1   Experimental methodology

To ensure each component is evaluated individually by the intended target users, we evaluate the more generative components by collecting feedback on their use from domain experts and ontology engineers through questionnaires. All questions ask participants to quantify the agreement with the statement made from 1 (strongly disagree) to 5 (strongly agree), with 3 being a neutral response (NR). No personal information is collected throughout the evaluation. The ontology testing feature, instead, is evaluated experimentally for accuracy.

User story questionnaire (domain experts). We recruited $N = 6$ music experts in the Polifonia project to create user stories summarising their requirements on music metadata using OntoChat. Our goal is to evaluate the model's success in producing satisfactory user stories. Additionally, we gather insights on OntoChat's usability and its effectiveness in minimising manual effort.

CQ extraction and clustering questionnaires (ontology engineers). We recruited $N = 8$ ontology engineers to evaluate the model's performance in generating CQs that are consistent with the given story, well-formulated, and consistent with the intended scope of the ontology. Users are first introduced to the concept of the user story, then asked to familiarise with the *Linka – Music Knowledge* story[9]. Their task is to extract and analyse CQs from the story using OntoChat.

Ontology testing evaluation. Given the OWL definition of Music Meta, and the full set of (manually annotated) CQ driving its implementation, we evaluate this component as a classification task. By extending this set with adversarial CQs, we expect OntoChat to correctly discriminate between these groups.

---

[8] `https://huggingface.co/spaces/b289zhan/OntoChat`
[9] `https://github.com/polifonia-project/stories/tree/main/Linka_Computer_Scientist`

4.2   Preliminary results

Feedback collected from domain experts confirmed that the user stories generated with OntoChat captured the intended goal and requirements and always provided relevant information (Fig. 3). More than 80% of participants enjoyed using the tool and found the final stories well-structured and easily understandable. While users acknowledged the model's ability to improve intermediate drafts through their feedback, only 50% were satisfied with the example data generated. Overall, 4/6 experts recognised the tool's potential to accelerate this task (2 NR), and 5 of them would prefer it over manual curation.

From the evaluation with ontology engineers, OntoChat was found to generate CQs that are comprehensive, reflective of the intended ontology scope, and easy to understand (Fig. 4). However, 2/8 participants noted the extraction of entities outside the story's scope; and only 50% observed the potential to reduce possible author bias. While 6/8 participants expressed satisfaction with OntoChat and recognised its time-saving benefits, all agreed it holds promise for streamlining CQ generation, indicating a preference over fully manual creation.

The clustering feature proved advantageous for understanding and organising ontology requirements when compared to full manual inspection (Fig. 5). Participants found the interaction intuitive (62.5%) and the resulting clusters expressed meaningful groupings of CQs (87.5%). While the feature offers time-saving benefits by providing an aggregated view of ontology requirements, there were indications that it may not fully support comprehensive analysis on its own.

Finally, by extending the original 28 CQs with the same number of negative CQs (expressing requirements that are not yet supported by Music Meta), we found that OntoChat can correctly classify supported requirements with an accuracy of 87.5% (P=88%, R=85.7%). More details are provided in Appx. B.4.

5   Conclusions

Ontology engineering in large collaborative projects faces challenges in managing complex interactions between stakeholders, domain experts, and ontology designers, leading to potential bottlenecks and inconsistencies in the resulting ontologies. After collecting strong evidence on the activities that are the most in need of computational support, we designed OntoChat, a framework implementing a conversational approach to ontology engineering, leveraging LLMs for requirement elicitation, analysis, and ontology testing. Evaluation results indicate positive reception among both domain experts and ontology engineers, with the framework seen as a tool that could significantly speed up traditional ontology engineering activities. Nonetheless, areas for improvement were also identified, such as enhancing the generation of data examples in user stories, refining name entity scope when creating CQs, and extending support for CQ analysis. While future work will focus on improving the framework based on the evaluation feedback and conducting a more comprehensive evaluation, this research underscores the potential of LLMs and conversational interfaces to enhance efficiency and collaboration in ontology engineering.

References

1. Aharoni, R., Goldberg, Y.: Unsupervised domain clusters in pretrained language models. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 7747–7763. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.692
2. Alharbi, R., Tamma, V., Grasso, F., Payne, T.: An Experiment in Retrofitting Competency Questions for Existing Ontologies. arXiv preprint arXiv:2311.05662 (2023)
3. Allen, B.P., Stork, L., Groth, P.: Knowledge Engineering using Large Language Models. arXiv preprint arXiv:2310.00637 (2023)
4. Babaei Giglou, H., D'Souza, J., Auer, S.: LLMs4OL: Large language models for ontology learning. In: International Semantic Web Conference. pp. 408–427. Springer (2023)
5. de Berardinis, J., Carriero, V.A., Jain, N., Lazzari, N., Meroño-Peñuela, A., Poltronieri, A., Presutti, V.: The Polifonia Ontology Network: Building a Semantic Backbone for Musical Heritage. In: Proceedings of the 22nd International Semantic Web Conference (ISWC) (2023)
6. de Berardinis, J., Carriero, V.A., Meroño-Peñuela, A., Poltronieri, A., Presutti, V.: The Music Meta Ontology: a flexible semantic model for the interoperability of music metadata. In: Proceedings of the the 24th International Society for Music Information Retrieval Conference (2023)
7. de Berardinis, J., Peñuela, A.M., Jain, N., Poltronieri, A., Lazzari, N., Presutti, V., Rigaux, P., Marzi, E., Graciotti, A., Wigham, M., Tirado, A.M., Gurrieri, M., Carriero, V.A., van Kranenburg, P.: Ontologies and knowledge graphs of music objects, patterns, and software package – 2nd version. Tech. rep., European Commission, The Polifonia consortium (2023)
8. Blomqvist, E., Hammar, K., Presutti, V.: Engineering Ontologies with Patterns - The eXtreme Design Methodology. In: Ontology Engineering with Ontology Design Patterns - Foundations and Applications, Studies on the Semantic Web, vol. 25. IOS Press, Amsterdam (2016). https://doi.org/10.3233/978-1-61499-676-7-23
9. Blomqvist, E., Presutti, V., Daga, E., Gangemi, A.: Experimenting with eXtreme Design. In: Knowledge Engineering and Management by the Masses. EKAW 2010. vol. 6317, pp. 120–134. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16438-5_9
10. Blomqvist, E., Seil Sepour, A., Presutti, V.: Ontology testing-methodology and tool. In: Knowledge Engineering and Knowledge Management: 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings 18. pp. 216–226. Springer (2012)
11. Bottini, T., Carriero, V.A., Carvalho, J., Cathé, P., Ciroku, F., Daga, E., Daquino, M., Davy-Rigaux, A., Guillotel-Nothmann, Gurrieri, M., van Kemenade, P., Marzi, E., Meroño Peñuelala, A., Mulholland, P., Musumeci, E., Presutti, V., Scharnhorst, A.: D1.1 Roadmap and pilot requirements 1st version. Tech. rep., EU Commission, The Polifonia consortium (2021)
12. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
13. Ciroku, F., de Berardinis, J., Kim, J., Meroño-Peñuela, A., Presutti, V., Simperl, E.: RevOnt: Reverse Engineering of Competency Questions from Knowledge Graphs via Language Models. Manuscript under review (2024)

14. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In: AAAI Conference on Artificial Intelligence (1997)
15. Funk, M., Hosemann, S., Jung, J.C., Lutz, C.: Towards Ontology Construction with Language Models. arXiv preprint arXiv:2309.09898 (2023)
16. Gruninger, M., Fox, M.S.: The role of competency questions in enterprise engineering. In: Benchmarking — Theory and Practice. IFIP Advances in Information and Communication Technology. pp. 83–95. Springer, Boston, MA (1994)
17. He, Y., Chen, J., Dong, H., Horrocks, I.: Exploring large language models for ontology alignment. arXiv preprint arXiv:2309.07172 (2023)
18. Hertling, S., Paulheim, H.: Olala: Ontology matching with large language models. In: Proceedings of the 12th Knowledge Capture Conference 2023. pp. 131–139 (2023)
19. Kendall, E.F., McGuinness, D.L.: Ontology engineering. Morgan & Claypool Publishers (2019)
20. Mateiu, P., Groza, A.: Ontology engineering with large language models. arXiv preprint arXiv:2307.16699 (2023)
21. Meyer, L.P., Stadler, C., Frey, J., Radtke, N., Junghanns, K., Meissner, R., Dziwis, G., Bulert, K., Martin, M.: Llm-assisted knowledge graph engineering: Experiments with chatgpt. arXiv preprint arXiv:2307.06917 (2023)
22. Peroni, S.: A simplified agile methodology for ontology development. In: OWL: Experiences and Directions–Reasoner Evaluation, pp. 55–69. Springer (2016)
23. Pinto, H.S., Staab, S., Tempich, C.: DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In: ECAI. vol. 16, p. 393. Citeseer (2004)
24. Qiang, Z., Wang, W., Taylor, K.: Agent-OM: Leveraging Large Language Models for Ontology Matching. arXiv preprint arXiv:2312.00326 (2023)
25. Simperl, E., Luczak-Rösch, M.: Collaborative ontology engineering: a survey. The Knowledge Engineering Review **29**(1), 101–131 (2014)
26. Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernández-López, M.: The NeOn methodology for ontology engineering. In: Ontology engineering in a networked world, pp. 9–34. Springer (2012)
27. Uschold, M., King, M.: Towards a methodology for building ontologies. Citeseer (1995)
28. Viswanathan, V., Gashteovski, K., Lawrence, C., Wu, T., Neubig, G.: Large Language Models Enable Few-Shot Clustering. arXiv preprint arXiv:2307.00524 (2023)
29. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (sep 2014). https://doi.org/10.1145/2629489, `https://doi.org/10.1145/2629489`
30. Zhang, B., Reklos, I., Jain, N., Peñuela, A.M., Simperl, E.: Using Large Language Models for Knowledge Engineering (LLMKE): A Case Study on Wikidata. arXiv preprint arXiv:2309.08491 (2023)
31. Zhang, Y., Wang, Z., Shang, J.: ClusterLLM: Large Language Models as a Guide for Text Clustering. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 13903–13920. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.emnlp-main.858

A   Survey results

As outline in Section 3, the design of OntoChat relies on our first research question presented in the introduction and recapitulated as follows: **RQ1** Which ontology engineering activities are the most in need of computational support?

To address RQ1, we run an online survey to collect feedback from ontology engineers. Based on their experience with OE tasks, they were asked to express their agreement with our statements. The latter were designed in order to understand the perceived complexity and the need of computational support for OE activities. Our survey received $N = 23$ responses.

The majority of participants (60.8%) declared to have strong knowledge and expertise in Ontology Engineering (OE). Participants have formerly used the following OE methodologies: NEON (47.4%), Ontology 101 (47.4%), eXtreme Design (42.1%), Linked Open Terms (42.1%), METHONDOLOGY (31.6%), and SAMOD (5.3%). They have experience working an OE projects with ontology design teams of various size: **1** (23.8%), **2-3** (47.6%), **4-6** (33.3%), and **7+** (14.3%). Analogously, they have worked on OE projects involving teams of stakeholders and domain experts of the various size: **1** (15%), **2-3** (45%), **4-6** (40%), and **7+** (25%). The results of this survey are detailed in Fig. 2 for consultation.

B   Evaluation results

B.1   Collaborative user story generation

This section provides more details on the evaluation of the *collaborative user story generation* feature in OntoChat. As outlined in Section 4, this evaluation was carried out by $N = 6$ domain experts who are familiar with the creation of user stories, and were actively involved in the OE activities behind of the Polifonia Ontology Network [5]. In line with our expectations, all participants confirmed their expert knowledge in the music (metadata) domain, and familiarity with the Music Meta Ontology. In addition, 50% of them acknowledge *some knowledge* of OE and have experience with the eXtreme Design methodology. The results of the evaluation are outlined in Fig. 3 for all the 10 questions.

B.2   Competency question extraction

Here, we provide more details on the evaluation of the *competency question extraction* feature in OntoChat. As outlined in Section 4, this evaluation was carried out by $N = 8$ participants with expertise in ontology engineering. The results of the evaluation are outlined in Fig. 4 for all the 8 questions.

B.3   Competency question clustering

As this evaluation step was performed after Competency Question Extraction by the same participants, their background information is the same as reported Appx. B.2. The results of the evaluation are outlined in Fig. 5.
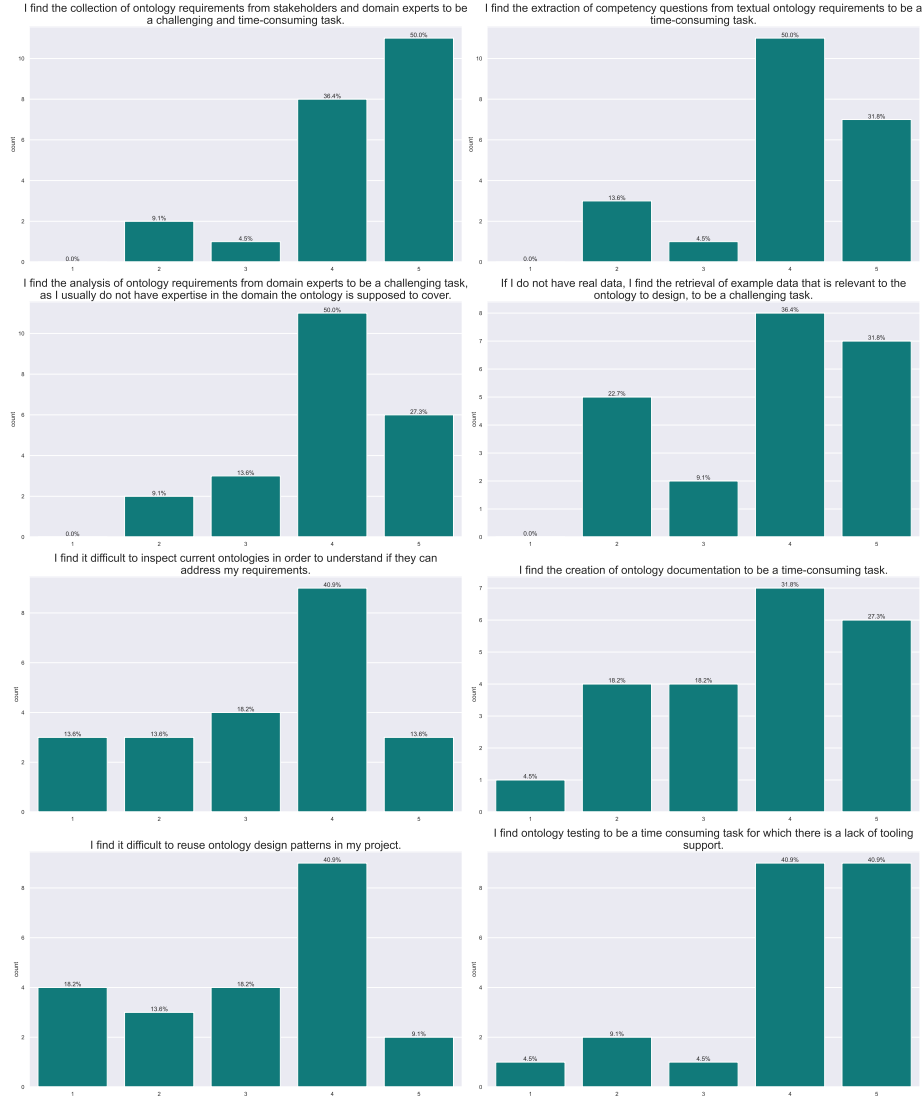
Fig. 2: Responses from the Ontology Engineering survey to address RQ1. Replies quantify the agreement of participants with respect to each statement on a 5-point Likert scale, where 1 (absolutely disagree) to 5 (absolutely agree), with 3 being a neutral response (neither agree nor disagree).
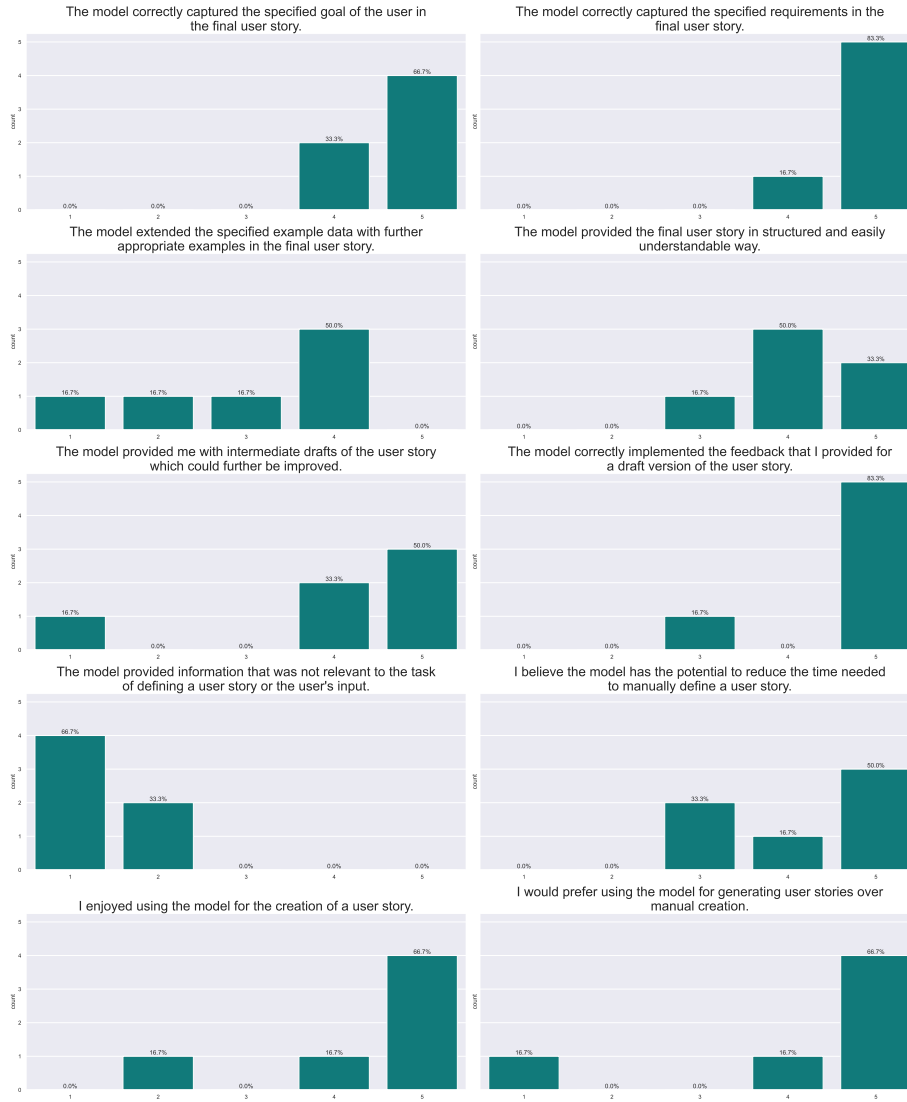
Fig. 3: User ratings on the Collaborative User Story Creation functionality of OntoChat. The evaluation was performed by $N = 6$ domain experts.

The model extracts competency questions representing the intended ontology scope.

The model writes competency questions that are comprehensive in relation to the given story.

The competency questions that the model creates are easy to understand.

The model did not provide competency questions with entities that are outside the scope of the user story.

I believe the model has the potential to reduce the time needed to manually extract competency questions from the user stories.

I believe the model has the potential to reduce possible author bias in the competency questions.

I enjoyed using the model for the creation of competency questions.

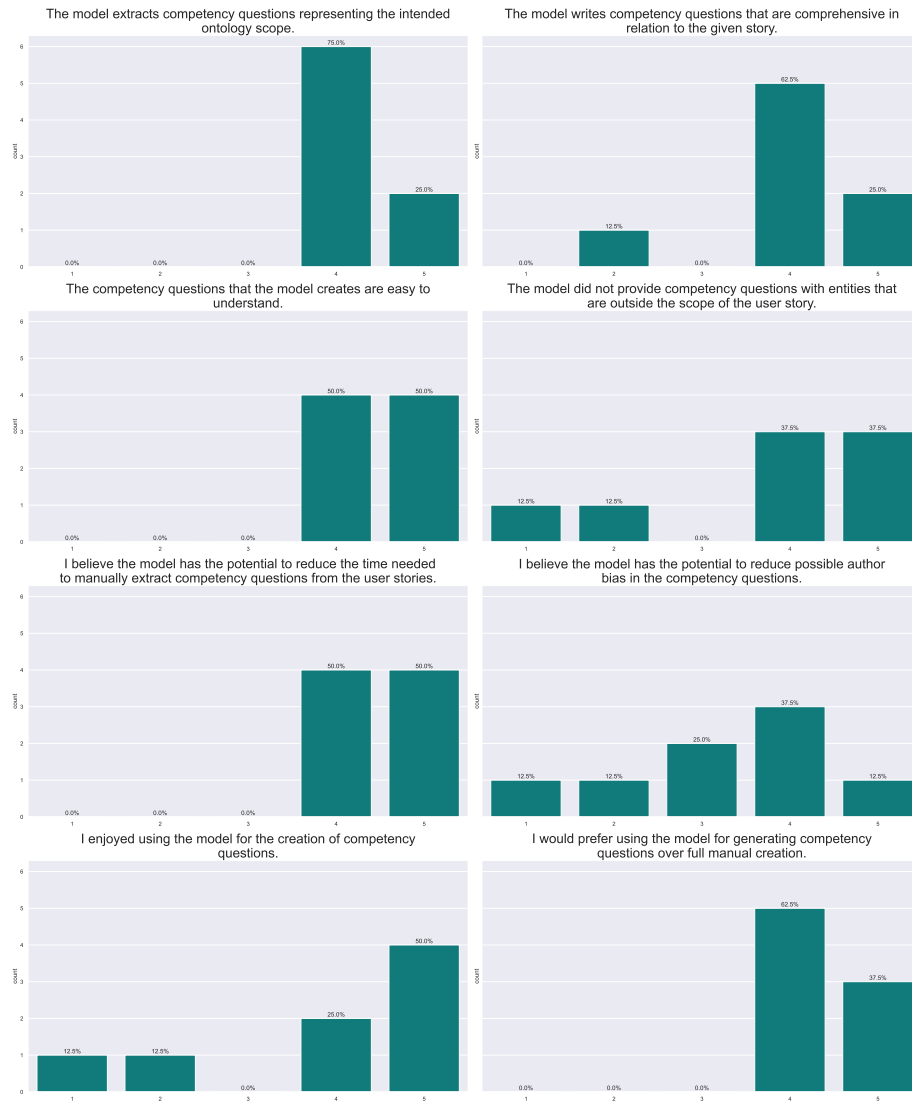I would prefer using the model for generating competency questions over full manual creation.

Fig. 4: User ratings on the *Competency Question Extraction* functionality of OntoChat. This evaluation was performed by $N = 8$ ontology engineers.
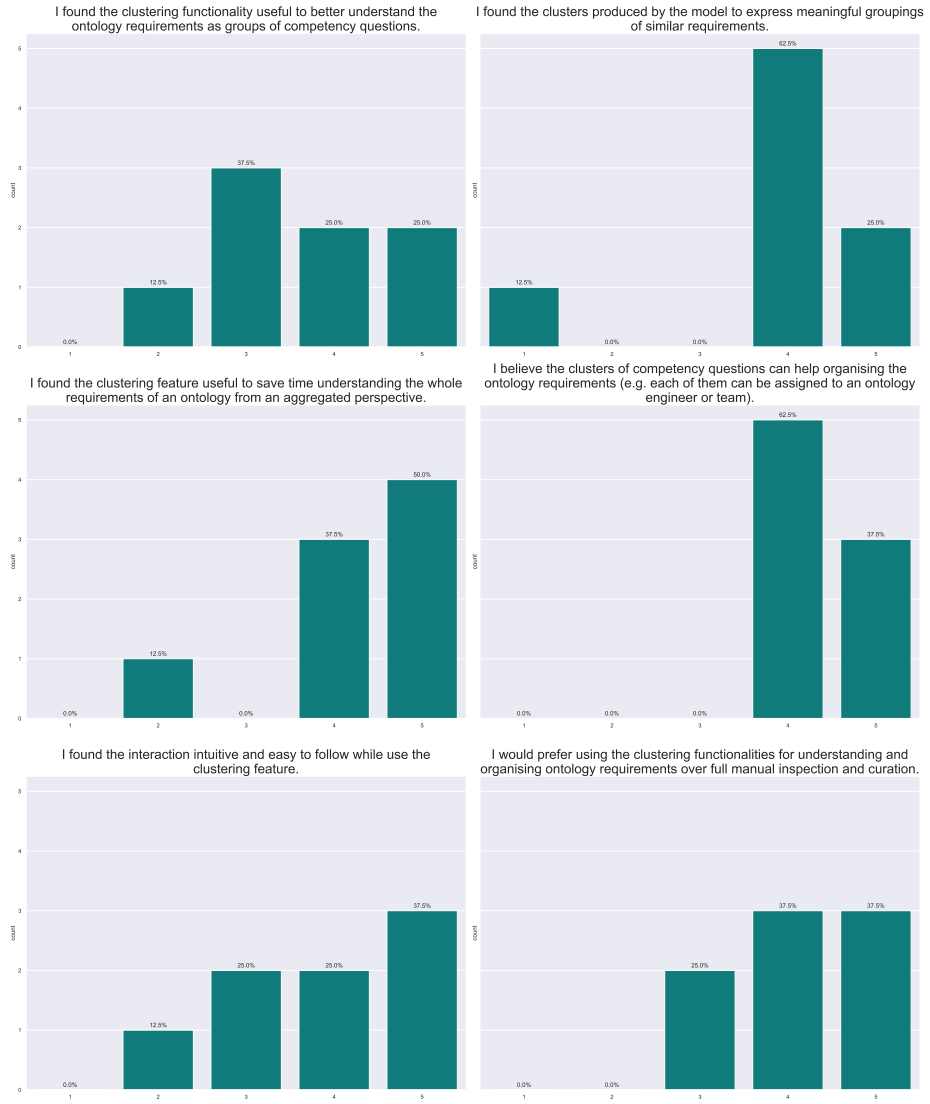
Fig. 5: User ratings on the Competency Question Clustering functionality performed by $N = 8$ ontology engineers.

B.4   Preliminary ontology testing

To complement our results for ontology testing, we report the confusion matrix in Fig. 6, expressing the number of correct predictions (25 true positives, 24 true negatives) and wrongly classified competency questions (3 false positives, 4 false negatives). Please, note that this can be seen as instance of competency question verification and error provocation for positive and negative CQs, respectively.
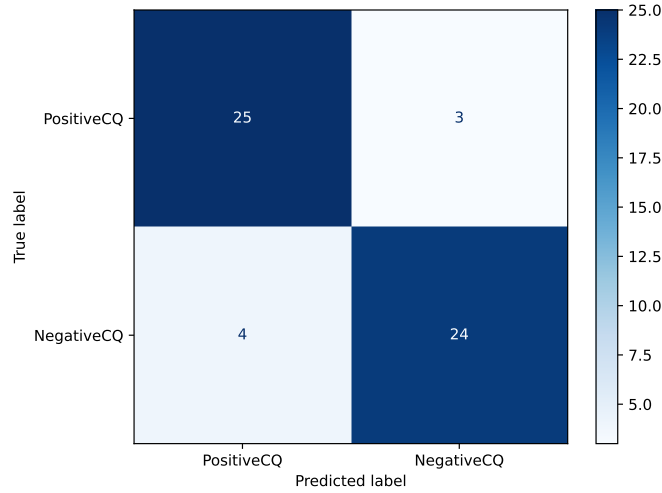
Fig. 6: Confusion matrix summarising our results for *prompt-driven CQ unit testing* (c.f. Section 3.4). Positive CQs (target label 1) denote competency questions that are expected to be addressed or covered by the ontology, whereas negative CQs (target label 0) are artificially created to express requirements that are not yet addressed by the ontology and should thus be predicted as such.

C   Examples

C.1   Collaborative user story generation

In this section, we provide several concrete example excerpts from the user story generation supported by OntoChat. Fig. 7 provides outputs of the knowledge elicitation stage while story refinement examples are visualised in Fig. 8. A full user story created with the assistance of OntoChat is shown in Fig. 9.

Hello! I am OntoChat, your conversational ontology engineering assistant. I will guide you step by step in the creation of a user story. Let's start with the persona. What are the name, occupations, skills, interests of the user?

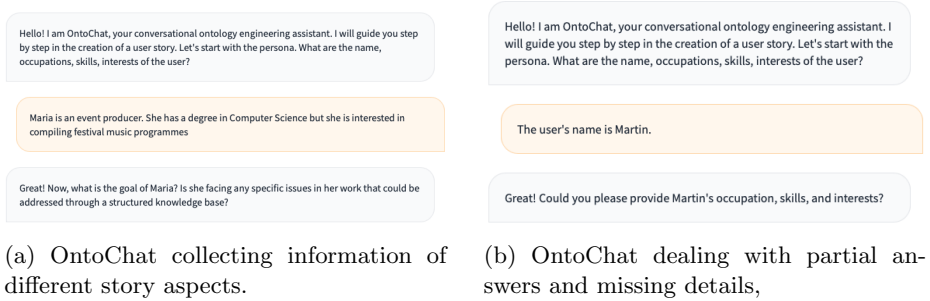Maria is an event producer. She has a degree in Computer Science but she is interested in compiling festival music programmes

Great! Now, what is the goal of Maria? Is she facing any specific issues in her work that could be addressed through a structured knowledge base?

Hello! I am OntoChat, your conversational ontology engineering assistant. I will guide you step by step in the creation of a user story. Let's start with the persona. What are the name, occupations, skills, interests of the user?

The user's name is Martin.

Great! Could you please provide Martin's occupation, skills, and interests?

(a) OntoChat collecting information of different story aspects.

(b) OntoChat dealing with partial answers and missing details,

Fig. 7: Example excerpts from the knowledge elicitation stage of the user story collaborative generation.

Feel free to provide any feedback or additional details to further refine the user story for Martin's needs.

I just remembered Martin also has a degree in Computer Science

Thank you for the additional information about Martin's background in Computer Science. Here is the updated user story:

Title: Music Program Compilation for Music Festivals
Persona: Martin, Event and Music Producer with a Degree in Computer Science

Please let me know if you would like to add more details or if you have any other feedback to improve the user story.

What I meant is that Maria is pursuing a Computer Science degree, she has not graduated yet

Title: Music Festival Program Compilation Assistance
Persona: Maria, Event Producer pursuing a degree in Computer Science and interested in compiling festival music programs

(a) Refinement by additional information.

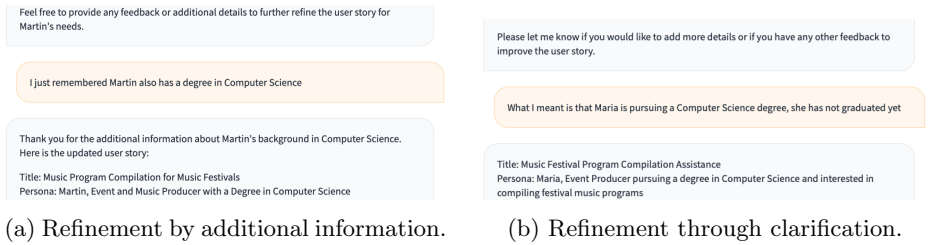(b) Refinement through clarification.

Fig. 8: Example excerpts from the story refinement stage of the user story collaborative generation.

User story

Title: Music Festival Program Compilation Assistance
Persona: Maria, Event Producer pursuing a degree in Computer Science and interested in compiling festival music programs
Goal: Maria aims to create cohesive and engaging festival music programs, leveraging her skill of selecting music pieces based on the current situation, but faces challenges in identifying relationships between different artists and songs.
Scenario: Maria, a student of Computer Science, is tasked with curating a music program for an upcoming festival. She needs to ensure a smooth flow between different artists and songs to create an enjoyable experience for the audience. Maria's ability to choose music based on the current situation is crucial. By using a structured knowledge base, Maria can access information on artists, songs, producers, and songwriters, helping her to make informed decisions on the sequencing and selection of music for the program, aligning with the event's atmosphere and audience preferences.
Example Data:
- Artist: Beyoncé
  - Songs: "Crazy in Love", "Single Ladies"
  - Producer: Timbaland
  - Songwriter: Beyoncé, Rich Harrison
- Artist: Ed Sheeran
  - Songs: "Shape of You", "Thinking Out Loud"
  - Producer: Benny Blanco
  - Songwriter: Ed Sheeran, Johnny McDaid

Let me know if you would like any further adjustments or additional details to be included in the user story.

Fig. 9: A user story generated with the assistance of OntoChat.

C.2   Competency question extraction

In this section, we provide examples from the *Competency Questions Extraction* functionality supported by OntoChat. Fig. 10 shows the first iteration of CQ extraction from a user story by OntoChat, while Fig. 11 illustrates the outputs after refinement (c.f. Section 3.2).
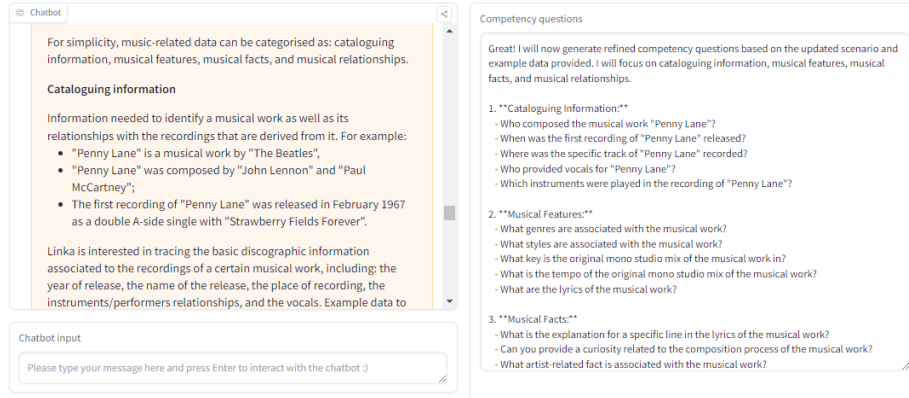

Fig. 10: Illustration of OntoChat's *competency question extraction* feature, first extraction of CQs.
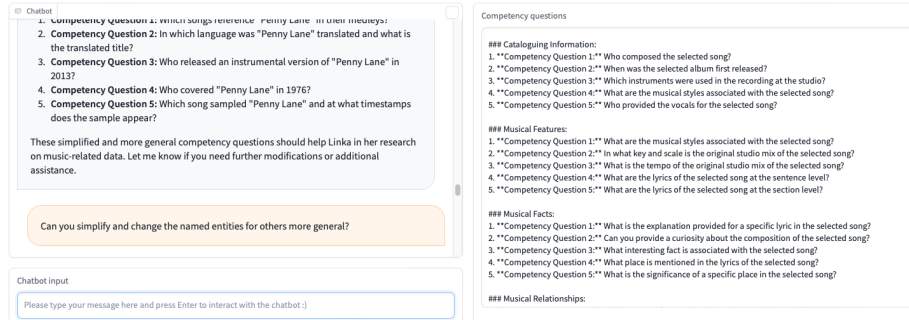

Fig. 11: Illustration of OntoChat's *competency question extraction* feature, refinement of the CQs.

C.3   Competency question clustering

We report an example of competency question clustering obtained after using this features on a selection of the original CQs driving the design of the Music

Meta Ontology [6], which can be fully found online[10]. The clusters produced by OntoChat via the LLM Clustering option (no number of clusters is provided apriori), together with their label, are reported in Listing 1.1.

```
"Music Artists": [
    "Which is the name of a music artist?",
    "Which is the alias of a music artist?",
    "Which is the language of the name/alias of a music artist?",
    "Which award was a music artist nominated for?",
    "Which award was received by a music artist?",
    "Which music artists has a music artist been influenced by?",
    "Which music artist has a music artist collaborated with?",
    "Which is the start date of the activity of a music artist?",
    "Which is the end date of the activity of a music artist?"
],
"Musical Pieces and Composers": [
    "Which is the composer of a musical piece?",
    "Is the composer of a musical piece known?",
    "In which time interval did the creation process took place?",
    "Where did the creation process took place?",
    "Which task was executed by a creative action?",
    "Which are the parts of a musical piece?",
    "Which collection is a musical piece member of?"
],
"Music Ensembles": [
    "Which are the members of a music ensemble?",
    "Which role a music artist played within a music ensemble?",
    "Where was a music ensemble formed?"
],
"Musical Performances and Recordings": [
    "Where was a musical piece performed?",
    "When was a musical piece performed?",
    "Which music artists took part to a musical performance?",
    "Which is the recording process that recorded a musical performance?",
    "Which is the recording produced by a recording process?"
], ...
```

Listing 1.1: Example of CQ clusters generated from Music Meta's (human-annotated) competency questions using the LLM clustering option in OntoChat.

C.4   Competency question clustering

We provide a selection of tested competency questions, with expected and predicted outcome (Yes = 1 for supported CQs, and No = 0 for CQs that are not supported by the current definition of the Music Meta ontology). These are reported in Table 1, and are complemented by a supporting explanation informing the user on how the requirement can/cannot be addressed by the ontology (e.g. classes and relations that can be used.

---

[10] https://github.com/polifonia-project/music-meta-ontology

CQ	Which award was received by a music artist?
Test prediction	Yes
Expected test outcome	Yes (supported)
Explanation	The ontology includes the class "Award" and the relation "Received award" which connects a MusicArtist to the Award they have received. This allows for querying the ontology to find the award(s) received by a specific music artist.
CQ	In which time interval did the creation process took place?
Test outcome	
Expected test outcome	Yes (supported)
Explanation	The ontology described can address the competency question "In which time interval did the creation process take place?" The ontology includes classes and properties related to the creative process, such as CreativeProcess, CreativeAction, and AnnotationMethod. It also includes properties like "isDerivedFrom" and "Creates" that can be used to track the creation process and infer the time interval in which it took place.
CQ	Which is the recording process that recorded a musical performance?
Test prediction	Yes
Expected test outcome	Yes (supported)
Explanation	The ontology can address the competency question "Which is the recording process that recorded a musical performance?" The ontology includes the class "RecordingProcess" and the property "isRecordedBy" which connects a RecordingProcess to a MusicEntity or MusicalPerformance.
CQ	Does a music algorithm favor a specific genre?
Test prediction	Yes
Expected test outcome	No (not supported, but potentially plausible)
Explanation	The ontology includes the necessary classes and relations to represent music algorithms, music genres, and the relationship between them. By using the "isTrainedOn" and "isTrainedBy" relations, it is possible to infer whether a music algorithm has been trained on a specific music genre or if a music genre has influenced the training of a music algorithm.
CQ	Is a music work associated to any case of plagiarism?
Test prediction	No
Expected test outcome	No (not supported)
Explanation	The ontology focuses on describing music metadata related to artists, compositions, performances, and recordings, but it does not include specific concepts or relations related to plagiarism detection or identifying cases of plagiarism in music works. To address the competency question, additional concepts and relations related to plagiarism detection and analysis would need to be added to the ontology.
CQ	Which language is most used in a music artist's lyrics?
Test prediction	No
Expected test outcome	No (not supported)
Explanation	The ontology described does not include any classes or properties related to language or lyrics analysis. It focuses more on the structural and organisational aspects of music metadata, such as compositions, performances, recordings, and their relationships. To address the question of which language is most used in a music artist's lyrics, additional classes and properties related to language analysis and text processing would need to be included in the ontology.
CQ	When was the album first sold?
Test prediction	Yes
Expected test outcome	No (not supported)
Explanation	The ontology includes classes and properties related to albums, such as Release, Publication Situation, and Publisher. By using these classes and properties, it is possible to model the publication and release of albums, including the specific situation or context in which they are made available to the public. The ontology can capture information about the timing, location, and associated entities involved in the publication process, allowing for the inference of when the album was first sold.

Table 1: Examples of tested competency questions with predicted and expected outcome. These are complemented by a supporting explanation.