CoXQL: A Dataset for Parsing Explanation Requests in Conversational XAI Systems

Anonymous ACL submission

Abstract

Conversational explainable artificial intelligence (ConvXAI) systems based on large language models (LLMs) have garnered significant interest from the research community in natural language processing (NLP) and human-computer interaction (HCI). Such systems can provide answers to user questions about explanations in dialogues, have the potential to enhance users' comprehension and offer more information about the decision-making and generation processes of LLMs. Currently available ConvXAI systems are based on intent recognition rather than free chat, as this has been found to be more precise and reliable in identifying users' intentions. However, the recognition of intents still presents a challenge in the case of ConvXAI, since little training data exist and the domain is highly specific, as there is a broad range of XAI methods to map requests onto. In order to bridge this gap, we present CoXQL[1], the first dataset for user intent recognition in ConvXAI, covering 31 intents, seven of which require filling multiple slots. Subsequently, we enhance an existing parsing approach by incorporating template validations, and conduct an evaluation of several LLMs on CoXQL using different parsing strategies. We conclude that the improved parsing approach (MP+) surpasses the performance of previous approaches. We also discover that intents with multiple slots remain highly challenging for LLMs.

1 Introduction

There is an increasing number of XAI systems that include user interfaces, facilitating natural language interaction with users (Chromik and Butz, 2021; Bertrand et al., 2023). More recently, there has been a significant development in building ConvXAI systems (Lakkaraju et al., 2022), which are

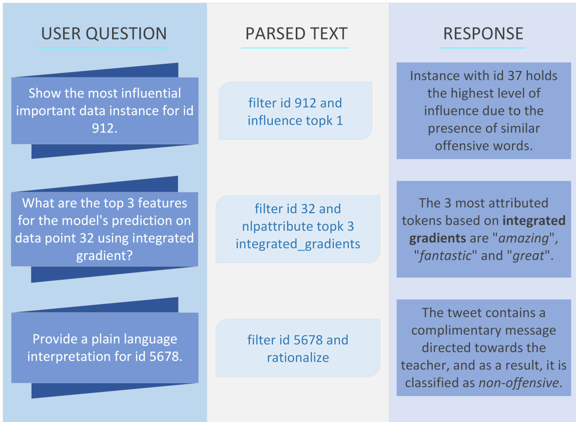

Figure 1: Example utterances consisting of user questions, SQL-like queries (parsed texts) and corresponding responses (not included in CoXQL) for influence (`influence`), feature attribution (`nlpattribute`) and rationalization (`rationalize`). More examples and operations can be found in Table 1 and Table 5.

guided through intent recognition rather than free-text chatting. The main reason for hard-coding intents is that in a ConvXAI application, there is a need for a maximally faithful conversation, which black-box generation cannot provide (Feldhus et al., 2023; Shen et al., 2023; Wang et al., 2024). These systems are designed to answer user questions about explainable language models in dialogues. In ConvXAI, intents usually represent the XAI operations supported in the system. The user experience and trust in the system can be negatively impacted when intent recognition fails (e.g., an incorrect mapping of XAI operations can lead to a discrepancy from users' requests). An extensive range of explainability questions has to be processed, which can be formulated in many different ways, depending on the domain of application (Lakkaraju et al., 2022). For instance, the user question: *"Clarify id 5678 with a reason."*, is formulated in different ways but represents the same rationalization intent as depicted in Figure 1.

[1]Conversational Explanation Query Language, a word play on CoSQL (Yu et al., 2019). Dataset and code are available at `https://anonymous.4open.science/r/CoXQL`.

	Operation	Description/Request
Loc.Pr.	predict(instance)	Get the prediction for the given instance
	likelihood(instance)	Calculate the model's confidence (or likelihood) on the given instance
Glob.Pr.	mistake({sample|count}, subset)	Count or show incorrectly predicted instances
	score(subset, metric)	Determine the relation between predictions and labels
Loc. Expl.	nlpattribute(inst., topk, method)	Provide feature attribution scores
	rationalize(inst.)	Explain the output/decision in natural language
	influence(inst., topk)	Provide the most influential training data instances
Pertrb.	cfe(instance)	Generate a counterfactual of the given instance
	adversarial(instance)	Generate an adversarial example based on the given instance
	augment(instance)	Generate a new instance based on the given instance
Data	show(instance)	Show the contents of an instance
	countdata(list)	Count instances
	label(dataset)	Describe the label distribution
	keywords(topk)	Show most common words
	similar(instance, topk)	Show most similar instances
Mod.	editlabel(instance)	Change the true/gold label of a given instance
	learn(instance)	Retrain or fine-tune the model based on a given instance
	unlearn(instance)	Remove or unlearn a given instance from the model
Meta	function()	Explain the functionality of the system
	tutorial(op_name)	Provide an explanation of the given operation
	data()	Show the metadata of the dataset
	model()	Show the metadata of the model
	domain(query)	Explain terminology or concepts outside of the system's functionality, but related to the domain

Table 1: Main operations in CoXQL as they can be requested in a dialogue (Description/Request), mapped onto a partial SQL-like query (Operation) that calls an explanation-generating or data-analyzing method. Red-highlighted operations are currently not implemented in any existing system. Additional logic operations are in Table 7.

In this work, we present the first dataset for explanation request parsing, CoXQL (§4). We frame the problem as a text-to-SQL-like task (§3.1). CoXQL consists of user questions and gold parses specifically designed for the XAI domain (Figure 1). It can serve as guidance for building ConvXAI systems and as a means to improve explanation intent recognition, where intents are considered as operations supported by ConvXAI systems. Moreover, we improve an existing parsing approach based on multi-prompt parsing (MP) (Wang et al., 2024) with additional template checks (§3.3) and find out that our improved approach (MP+) easily outperforms existing approaches. Lastly, we evaluate several state-of-the-art LLMs with various parsing strategies on CoXQL for explanation intent recognition (§5). Our evaluation shows that CoXQL can be regarded as a benchmark for future research and still presents challenges for state-of-the-art LLMs, especially for accurately recognizing intents (operations) with multiple slots, where slots are finer-grained user preferences regarding XAI operations (e.g., `topk` and `integrated gradient` associated with feature attribution in Figure 1).

2 Related Work

In the majority of previous ConvXAI systems (Werner, 2020; Nguyen et al., 2023; Shen et al., 2023), the semantic similarity of sentence embeddings between user query and existing data is used to match the user query with the appropriate operation (Table 3), known as the nearest neighbor. In contrast, the approach used in TalkToModel (Slack et al., 2023), InterroLang (Feldhus et al., 2023) and LLMCheckup (Wang et al., 2024) employs LLMs to convert user questions into SQL-like queries (Figure 1). The best performance is achieved in Slack et al. (2023), Feldhus et al. (2023) and Wang et al. (2024) with a fine-tuned `T5`, an adapter-based `BERT`, and `Llama2` with few-shot prompting, respectively. This parsing approach demonstrates notable enhancements, exceeding a doubling in parsing accuracy compared to the nearest neighbor approach. While they all support no more than 24 operations in their systems, CoXQL contains in total 31 operations of various complexity ranging from single term operations to operations with multiple slots.

3 Methodology

3.1 Task Framing

Building upon the strategy employed by Slack et al. (2023), Feldhus et al. (2023) and Wang et al. (2024) (§2), we treat XAI intent recognition as a text-to-SQL-like task (Figure 1), which can be effectively modeled as a seq2seq task (Sutskever et al., 2014).

The generated SQL-like queries should be correctly executable ensuring practical usability and functionality, since failed intent recognition results in incorrect XAI responses, leading to a negative impact on the user experience (Feldhus et al., 2023).

3.2 Supported Operations

We have determined **23** XAI and supplementary operations, which we show in Table 1, and **8** additional operations related to logic and filtering depicted in Table 7. The list of available operations (Table 1), including five newly introduced ones (marked in red in Table 1; App. I), are consolidated from HCI literature (Weld and Bansal, 2019; Liao et al., 2021), the state-of-the-art ConvXAI systems by Slack et al. (2023), Shen et al. (2023), Feldhus et al. (2023) and Wang et al. (2024), and the taxonomy for LLM interpretation research by Singh et al. (2024). Moreover, several operations (Table 6) are associated with multiple slots, which makes parsing even more challenging for LLMs (Table 10). The inclusion of additional fine-grained slots is favored in ConvXAI systems (e.g., `integrated gradient` in Figure 1), enabling the provision of more informative and multi-faceted explanations (Nobani et al., 2021; Wijekoon et al., 2024).

3.3 Parsing

Nearest Neighbor Nearest neighbor (NN) relies on comparing semantic similarity between user query and existing training samples measured by an SBERT model[2]. However, as the number of operations and additional slots (e.g., ranges of values, method names) associated with operations grow, the intent recognition accuracy tends to decrease.

Guided Decoding Guided Decoding (GD) relies on a predefined grammar to restrict the generated output of LLMs (Figure 4) (Shin et al., 2021). The parsing prompt used in GD consists of demonstrations that are selected based on their semantic similarity to the desired output (Table 4) (Slack et al., 2023).

Multi-prompt Parsing With GD, due to similarity-based pre-selection, the model might miss the demonstrations for the actual operation. Multi-prompt Parsing (MP) (Wang et al., 2024) first queries the model about the main operation by providing coarse-grained demonstrations for all available operations (Table 1) and then selects more fine-grained operation-specific prompts in the next step (Table 6).

[2]`https://huggingface.co/BAAI/bge-base-en-v1.5`

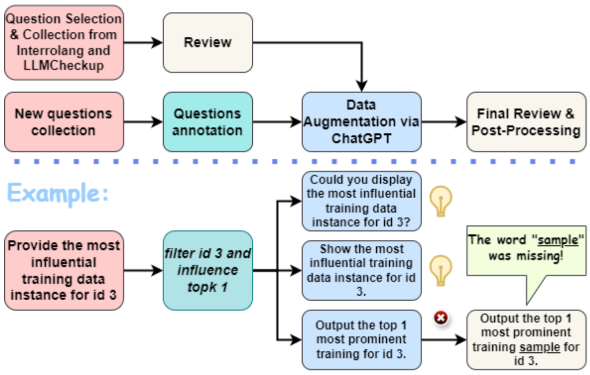

Figure 2: The data collection pipeline of CoXQL.

Multi-prompt Parsing with template checking Compared to GD, MP is not constrained by the grammar and the parsed text generated by MP is not guaranteed to adhere to the expected template (e.g., the exact order or naming of all slots; Table 10). We also find that extracting ids and numerical slots poses a challenge for out-of-the-box prompting with MP. Thus, we improve MP and introduce MP+ that uses additional template checking. This is an important step, since template checking contributes to more reliable parsing that takes both grammar and user input into account[3].

4 The CoXQL Dataset

4.1 Dataset Creation

The data creation process of CoXQL is depicted in Figure 2. Based on the predefined set of question and parse pairs from InterroLang (Feldhus et al., 2023) and LLMCheckup (Wang et al., 2024), we selectively choose[4] pairs of question and gold parse for operations marked in blue in Table 1. Meanwhile, we manually create new additional pairs for all operations in Table 1, following the way how questions are raised in Feldhus et al.'s (2023) user study. Subsequently, we use `ChatGPT` (OpenAI, 2022) to augment user questions (Figure 5) to expand the dataset size. The generated pairs undergo a review process and are post-processed by us if needed (e.g., adding missing words; Figure 2).

[3]More details about MP+ are in App. F.

[4]E.g., by evaluating questions' understandability or topic-parse alignment. More details are provided in App. H.

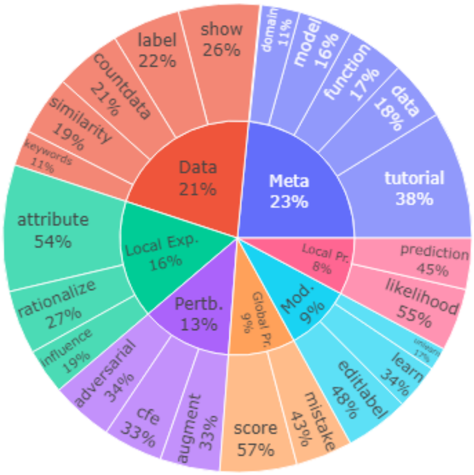

Figure 3: The intent distribution of CoXQL.

4.2 Data Statistics

After all processing steps, CoXQL comprises **1179** pairs of user questions and corresponding SQL-like queries over full SQL parses, 82 of which were post-processed manually. Figure 3 illustrates the intent distribution of CoXQL. Operations with additional slots in Table 6 have an intentionally higher number of instances compared to others due to their difficulty. Moreover, Table 5 provides examples of utterances along with their corresponding parses. Three authors of this work performed the annotations. We report a token-level inter-annotator agreement of Fleiss' $\kappa = 0.87$. While LLMs find it challenging to understand different formulations of XAI questions and recognize slots associated with operations simultaneously, these tasks are not as difficult for humans. In addition, we manually crafted **112** pairs specifically for the test set, which is evaluated in §5. More details about post-processing and test set are given in Appendix H.

5 Evaluation

To assess the ability of interpreting user intents with LLMs, we quantify the performance of seven LLMs[5] with different sizes ranging from 1B to 70B, employing four approaches: NN, GD, MP and MP+ (§3.3) (Table 2). Performance is calculated by measuring exact match parsing accuracy (Talmor et al., 2017; Yu et al., 2018) on CoXQL. We find that MP falls short of GD on CoXQL except `CodeQWen1.5` (Bai et al., 2023), while improved MP (MP+) can easily outperform GD and MP with additional template checks. Among all LLMs and parsing strategies, our findings reveal that `Llama3-70B` with MP+ demonstrates the highest scores, exhibiting a doubling in performance compared to the baseline (NN).

[5]Two of them, `CodeQWen` (Bai et al., 2023) and `sqlcoder`, are designed for code and SQL generation. Deployed LLMs are indicated in the left column of Table 2 and in Table 8.

Model	Size	NN	GD	MP	MP+
Baseline	-	44.25	-	-	-
`Falcon`	1B	-	59.29	59.29	77.88
`Pythia`	2.8B	-	79.65	74.34	83.19
`Mistral`	7B	-	78.76	78.76	87.61
`Llama3`	8B	-	84.07	67.26	86.73
`Llama3`	70B	-	83.19	68.14	93.81
`CodeQwen1.5`	7B	-	65.49	67.25	85.84
`sqlcoder`	7B	-	86.73	79.65	88.50

Table 2: Exact match parsing accuracy (in %) for different models on the CoXQL test set. NN = Nearest Neighbor; GD = Guided Decoding prompted by 20-shots; MP = Multi-prompt Parsing; MP+ = MP with template checks.

A detailed error analysis for each category is given in Table 9. GD outperforms MP when operations involve a greater number of additional slots (Table 6), which is due to MP's tendency to generate a higher volume of slots and MP not being constrained by grammar. Nevertheless, MP+ can achieve overall better results. Additionally, Table 10 shows parsed texts of the question: *"Top 3 important features for id 3!"*, generated by all deployed LLMs. None of them can fully match the gold parse, regardless of LLMs or parsing strategies, which demonstrates that LLMs still face great challenges when dealing with operations that involve multiple slots (Appendix J).

6 Conclusion

The contributions of this paper are three-fold: Firstly, we present and release the first dataset for explanation request parsing in ConvXAI with 31 intents, CoXQL. Secondly, we improve the previous parsing strategy MP with additional template checks, which considerably improves parsing accuracy. Lastly, we perform a comparative evaluation of seven state-of-the-art LLMs on the CoXQL data. We find that MP+ outperforms both GD and MP but LLMs still struggle with intents that have multiple slots. In the future, we would like to consider tools like LANGCHAIN[6] to provide more accessible, extensible framework.

[6]`https://www.langchain.com/`

Limitations

CoXQL currently supports only English, and it does not offer multilingual support. However, it is feasible to adapt CoXQL to target languages through translation.

The complexity of user questions in CoXQL might be lower when compared to other text-to-SQL datasets that involve complex SQL grammar, such as JOINs, aggregations. Within the current scope, we do not take into account the concatenation of various operations, which could potentially be valuable for users.

All implementations for operations shown in Table 1 highlighted in blue can be found in either TalkToModel (Slack et al., 2023), InterroLang (Feldhus et al., 2023) or LLMCheckup (Wang et al., 2024). CoXQL provides annotations for the ones highlighted in red in Table 1. Although none of the existing systems supports additional operations, they can be implemented as described in Appendix I.

While some LLMs, e.g. `Llama3-70B`, can achieve good results in explanation request parsing, their deployment may not always be feasible, e.g., due to resource limitations. This challenge can potentially be addressed by employing active learning techniques on smaller-sized LMs to attain comparable parsing accuracy.

References

Tanja Baeumel, Soniya Vijayakumar, Josef van Genabith, Guenter Neumann, and Simon Ostermann. 2023. Investigating the encoding of words in BERT's neurons using feature textualization. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 261–270, Singapore. Association for Computational Linguistics.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Astrid Bertrand, Tiphaine Viard, Rafik Belloum, James R. Eagan, and Winston Maxwell. 2023. On selective, mutable and dialogic xai: A review of what users say about different types of interactive explanations. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA. Association for Computing Machinery.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Michael Chromik and Andreas Butz. 2021. Human-XAI interaction: a review and design principles for explanation user interfaces. In *Human-Computer Interaction–INTERACT 2021: 18th IFIP TC 13 International Conference, Bari, Italy, August 30–September 3, 2021, Proceedings, Part II 18*, pages 619–640. Springer.

Nils Feldhus, Qianli Wang, Tatiana Anikina, Sahil Chopra, Cennet Oguz, and Sebastian Möller. 2023. InterroLang: Exploring NLP models and datasets through dialogue-based explanations. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5399–5421, Singapore. Association for Computational Linguistics.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, Online. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv*, abs/2310.06825.

Himabindu Lakkaraju, Dylan Slack, Yuxin Chen, Chenhao Tan, and Sameer Singh. 2022. Rethinking explainability as a dialogue: A practitioner's perspective. *HCAI @ NeurIPS 2022*.

Dong-Ho Lee, Akshen Kadakia, Brihi Joshi, Aaron Chan, Ziyi Liu, Kiran Narahari, Takashi Shibuya, Ryosuke Mitani, Toshiyuki Sekiya, Jay Pujara, and Xiang Ren. 2023. XMD: An end-to-end framework for interactive explanation-based debugging of NLP models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 264–273, Toronto, Canada. Association for Computational Linguistics.

Piyawat Lertvittayakumjorn and Francesca Toni. 2021. Explanation-Based Human Debugging of NLP Models: A Survey. *Transactions of the Association for Computational Linguistics*, 9:1508–1528.

Q. Vera Liao, Milena Pribic, Jaesik Han, Sarah Miller, and Daby Sow. 2021. Question-driven design process for explainable AI user experiences. *arXiv*, abs/2104.03483.

Van Bach Nguyen, Jörg Schlötterer, and Christin Seifert. 2023. From black boxes to conversations: Incorporating XAI in a conversational agent. In *Explainable Artificial Intelligence*, pages 71–96, Cham. Springer Nature Switzerland.

Navid Nobani, Fabio Mercorio, and Mario Mezzanzanica. 2021. Towards an explainer-agnostic conversational xai. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4909–4910. International Joint Conferences on Artificial Intelligence Organization. Doctoral Consortium.

OpenAI. 2022. Introducing ChatGPT. `https://openai.com/blog/chatgpt/`.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data only. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc.

Hua Shen, Chieh-Yang Huang, Tongshuang Wu, and Ting-Hao Kenneth Huang. 2023. ConvXAI: Delivering heterogeneous AI explanations via conversations to support human-AI scientific writing. In *Computer Supported Cooperative Work and Social Computing*, CSCW '23 Companion, page 384–387, New York, NY, USA. Association for Computing Machinery.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chandan Singh, Jeevana Priya Inala, Michel Galley, Rich Caruana, and Jianfeng Gao. 2024. Rethinking interpretability in the era of large language models. *arXiv*, abs/2402.01761.

Dylan Slack, Satyapriya Krishna, Himabindu Lakkaraju, and Sameer Singh. 2023. Explaining machine learning models with interactive natural language conversations using TalkToModel. *Nature Machine Intelligence*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Alon Talmor, Mor Geva, and Jonathan Berant. 2017. Evaluating semantic parsing against a simple web-based question answering model. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 161–167, Vancouver, Canada. Association for Computational Linguistics.

Vittorio Torri. 2021. Textual eXplanations for intuitive machine learning. Master's thesis, Politecnico di Milano, dec.

Igor Tufanov, Karen Hambardzumyan, Javier Ferrando, and Elena Voita. 2024. LM transparency tool: Interactive tool for analyzing transformer language models. *arXiv*, abs/2404.07004.

Qianli Wang, Tatiana Anikina, Nils Feldhus, Josef van Genabith, Leonhard Hennig, and Sebastian Möller. 2024. LLMCheckup: Conversational examination of large language models via interpretability tools. *arXiv*, abs/2401.12576.

Daniel S. Weld and Gagan Bansal. 2019. The challenge of crafting intelligible intelligence. *Commun. ACM*, 62(6):70–79.

Christian Werner. 2020. Explainable ai through rule-based interactive conversation. In *Proceedings of the Workshops of the EDBT/ICDT 2020 Joint Conference*.

Anjana Wijekoon, David Corsar, Nirmalie Wiratunga, Kyle Martin, and Pedram Salimi. 2024. Tell me more: Intent fulfilment framework for enhancing user experiences in conversational xai.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

A Approaches for intent recognition

Table 3 displays the approaches for intent recognition in the current XAI systems.

XAI System Implementations	Intent recognition: Embeds	Fine-Tuned	Few-Shot	Text-to-SQL
Werner (2020)	fastText			
Torri (2021)		`GPT-2`		
Slack et al. (2023)	`MPNet`	`T5`	`GPT-J`	■
Nguyen et al. (2023)	`SimCSE`			
Shen et al. (2023)	`SciBERT`			
Feldhus et al. (2023)	`MPNet`	`BERT+Adap`, `FLAN-T5`	`GPT-Neo`	■
Wang et al. (2024)	`MPNet`		`Llama2`	■
Ours	`bge-base`		`Llama3`	■

Table 3: Approaches for intent recognition in conversational XAI systems using LM embeddings, fine-tuned LMs and LLMs with few-shot prompting.

B Guided decoding

B.1 Example grammar

Figure 4 shows the grammar for `mistake` operation with additional slots `count` and `sample`.

B.2 Demonstration selection

As described in §3.3, for guided decoding, the parsing prompt will contain demonstrations which are selected based semantic similarity. Table 4 shows the top 3 similar selected demonstrations for the user question "Can you show me how much data the model predicts incorrectly?".

C Example utterance from CoXQL

Table 5 provides example utterances corresponding to each operation listed in Table 1.

D Additional slots for operations

Table 6 shows operations with additional slots.

E Filter and logic operations

In addition to the operations displayed in Table 1, we have also incorporated operations related to logic and filtering, as depicted in Table 7. While InterroLang (Feldhus et al., 2023) and LLMCheckup (Wang et al., 2024) already include `predfilter`, `labelfilter` and `previousfilter`, we introduce a new filter called `lengthfilter`, which allows for dataset filtering based on the length of the instances at various levels of granularity, such as character, token, or sentence.

Those aforementioned filters allows for a wide range of possibilities in analyzing and manipulating the dataset based on various conditions and interests. For instance, one can examine data points where the predicted label differs from the golden label using a combination of `labelfilter` and `predfilter`. In addition, all filters can be interconnected with operations listed in Table 1.

F Multi-prompt parsing

As indicated in Section 3.3, MP is not constraint by the predefined grammar. From Table 9, we found that extracting ids and numerical slots poses a significant challenge for out-of-the-box prompting, especially for those LLMs that have less parameters (e.g., `falcon-1B` or `Pythia-2.8B`). Vanilla MP shows lower performance on operations from Table 6 that require several slots (e.g., `Global Prediction` and `Local Explanation`, see Table 9). The lower performance of MP compared to GD can be attributed to the fact that MP tends to generate a larger volume of tokens/slots, given MP's lack of constraints imposed by grammar. For instance, in the case of `score` operation, which can take values such as `accuracy`, `precision`, `roc`, `recall`, or `f1` as additional slots, MP has a tendency to produce more than one metric name. Thus, we propose MP+, which applies additional template checks on the generated parsed text and can achieve best performance compared to GD and MP (§5).

G Prompt design

Figure 5 shows the prompt used with `ChatGPT` to produce additional data points for CoXQL.

H Data collection

Data collection pipeline We employ a selective approach where we choose question and parse pairs from InterroLang (Feldhus et al., 2023) and LLMCheckup (Wang et al., 2024) specifically for operations that are also present in CoXQL. Subsequently, we thoroughly review all the collected user

```
GRAMMAR = r"""
?start: mistake
mistake: mistakesword mistakestypes
mistakesword: " mistake"
mistakestypes: " count" | " sample"
"""
```

Figure 4: Example grammar of `mistake` operation with additional slots "count" and "sample".

Type	Text
User question	Can you show me how much data the model predicts incorrectly?
Selected demonstration	Tell me the amount of data the model predicts falsely. Can you demonstrate how many data points are predicted wrongly? Show me some data you predict incorrectly.

Table 4: Selected top 3 demonstrations based on semantic similarity.

	Intent class	Example utterance	Gold parse
Loc.Pr.	`predict`	What is the prediction for data point number 9130?	`filter id 9130 and predict`
	`likelihood`	Give me the confidence score for this prediction on id 15?	`filter id 15 and likelihood`
Glob.Pr.	`mistake`	Tell me the amount of data the model predicts falsely.	`mistake count`
	`score`	Give me the accuracy on the data.	`score accuracy`
Loc. Exp.	`attribute`	Why do you predict instance 2451?	`filter id 2451 and nlpattribute default`
	`rationalize`	Generate a natural language explanation for id 2222.	`filter id 2222 and rationalize`
	`influence`	Show the most influential important data instance for id 912.	`filter id 912 and influence topk 1`
Pertrb.	`cfe`	How would you flip the prediction for id 23?	`filter id 23 and cfe`
	`adversarial`	How would you construct an adversarial example for the model's prediction on id 23?	`filter id 23 and adversarial`
	`augment`	Can you modify and generate a new instance from id 100?	`filter id 100 and augment`
Data	`show`	Could you show me data point number 215?	`filter id 215 and show`
	`countdata`	Count the total number of data points.	`countdata`
	`label`	Please show what the gold labels are.	`label`
	`keywords`	What are the most frequent keywords in the data?	`keywords topk 1`
	`similar`	Is it possible to retrieve an example that is similar to id 12?	`filter id 12 and similarity topk 1`
Mod.	`editlabel`	Edit the label of id 2894 to the specified label.	`filter id 2894 and editlabel`
	`learn`	Apply training to the model using instance 473.	`filter id 473 and learn`
	`unlearn`	Can you unlearn id 530 from the model?	`filter id 530 and unlearn`
Meta	`function`	Tell me a bit more about what I can do here.	`function`
	`tutorial`	What's data augmentation?	`qatutorial qada`
	`data`	Tell me a bit more about the data please.	`data`
	`model`	It would be very useful if you could provide a description of the model!	`model`
	`domain`	Can you clarify terms or concepts that are relevant to the domain but not directly related to the system's functionality?	`domain`

Table 5: Intent classes, example utterance from CoXQL and corresponding gold parse.

```
system_prompt = (f"As an expert in data augmentation, you will involve receiving pairs of user questions and parsed text. Your task is to rephrase the user questions in a manner that preserves their semantic meaning while keeping the parsed text unchanged. Here are some examples.\n")

read_instruction = f"User question: {user_question}\n Parsed text: {parsed_text}\n"

# Combine inputs to single string
entire_prompt = system_prompt + demonstrations + read_instruction
```

Figure 5: Simplified version of the Python code showing the data augmentation prompt using `ChatGPT` to generate additional data points for CoXQL.

Operation	Additional Slots	#Additional Slots
`influence`	*topk*	1
`keywords`	*topk*	1
`similarity`	*topk*	1
`mistake`	sample, count	2
`score`	accuracy, precision, recall, f_1, roc	5
`attribute`	all, *topk*, default attention, lime, integrated gradient, inputxgradient	7
`tutorial`	qaattribute, qarationalize, qainfluence, qacfe qaadversarial, qaaugment, qaeditlabel, qalearn, qaunlearn	9

Table 6: Additional slots for operations.

	Operation	Description/Request
Filter	`filter(id)`	Access single instance by its ID
	`predfilter(label)`	Filter the dataset according to the model's predicted label
	`labelfilter(label)`	Filter the dataset according to the true/gold label given by the dataset
	`lengthfilter(level, len)`	Filter the dataset by length of the instance (characters, tokens, . . .)
	`previousfilter()`	Filter the dataset according to outcome of previous operation
	`includes(token)`	Filter the dataset by token occurrence
Logic	`and(op1, op2)`	Concatenate multiple operations
	`or(op1, op2)`	Select multiple filters

Table 7: Additional logic operations in CoXQL.

questions, assessing aspects such as readability, understandability, and coherence. Additionally, we ensure that the purpose or topic conveyed within the user question aligns with the corresponding parse. If post-processing is required, such as in the case of pairs from INTERROLANG (Feldhus et al., 2023) and LLMCHECKUP (Wang et al., 2024), and pairs generated by `ChatGPT`, we may need to reformulate the user questions or potentially modify the parsed text based on the intended meaning or intent of the questions. E.g. when we use `ChatGPT` to augment the user question *"Why do you predict instance id 31 using input gradient?"*, which should be parsed as "`filter id 31 and nlpattribute all input_x_gradient`". Since `nlpattribute` operation (feature attribution) has many additional slots (Table 6), `ChatGPT` generates the parsed text of the mentioned question as "`filter id 31 and nlpattribute topk 1 input_x_gradient`" (the additional slot should be `all` instead of `topk 1` because user question does not specify the top k values and thus `all` should be set as default), although we instruct `ChatGPT` to not change the parsed text in the prompt (Figure 5). In such a case, we have to post-process the parsed text by changing "topk 1" to "all".

Test set creation Feldhus et al. (2023) conducts a user study to evaluate the quality of explanations generated by INTERROLANG. The user questions, along with their corresponding answers and parsed texts from this user study, are publicly accessible[7]. Inspired by Feldhus et al.'s (2023) approach, we adopt a similar strategy and a subset of the test set is created following the way how questions are raised from the user study.

I Operations not supported in current XAI dialogue systems

We introduce five new operations, which are currently not present in the existing ConvXAI systems outlined in Table 1 and Table 7 marked in red. `influence` operation enables the retrieval of the most influential training data contributing to the result (Han et al., 2020). `editlabel` operation allows for the modification of the golden label for

[7] https://github.com/DFKI-NLP/InterroLang/blob/main/feedback

a specific instance. With the `learn` and `unlearn` operations, the deployed model can be additionally fine-tuned with or without a particular instance. The `domain` operation provides information regarding terminology or concepts relevant to our domain but not covered by the system.

We outline here how we would implement them:

- `influence(instance, topk)`: To calculate influential training instances, CAPTUM provides a tutorial for the TracIn method: https://captum.ai/tutorials/TracInCP_Tutorial. However, it is quite expensive to execute on LLMs.
- Modification operations are related to explanatory debugging, an area of research surveyed in Lertvittayakumjorn and Toni (2021). A representative system is XMD (Lee et al., 2023).
- `domain(query)`: The entire user question is provided to the LLM and the operation is treated as an open-domain question answering task similar to the `rationalize` operation.
- `lengthfilter(level, len)` is straightforward to implement and only considers the dataset instances with a length above or below some character, token, word, or sentence count (specified by the granularity `level` slot).

Additionally, we want to point out that in practical applications of XAI systems, it is common to encounter a significant number of questions belonging to `domain` operation. In such cases, the TOOLFORMER (Schick et al., 2023) can be integrated and utilized to directly access relevant tools or APIs associated with the domain-specific questions.

CoXQL deliberately excluded attention head and circuit analyses (Baeumel et al., 2023) which are not well-suited for conversational explanations and are dependant on visualization rather than text as a modality for explanation. We propose to use dedicated tools for those purposes (Tufanov et al., 2024).

J Parsing accuracy evaluation

J.1 Models for parsing accuracy evaluation

Table 8 lists all LLMs that are evaluated for parsing. We used A100 and H100 for parsing accuracy evaluation, which is done within 1 hour per setting.

J.2 Error analysis

Error analysis at the category level Table 9 displays F_1 scores of each category for different LLMs shown in Table 8. From Table 9, we find out that GD generally performs better than MP in categories like `Global Prediction`, `Local Explanation`, and `Local Prediction`. MP, however, performs better in categories like `Data` and `Modification`. MP+ exceeds the performance of both GD and MP across most categories and models, indicating that the combination of Multi-Prompt parsing with template checks provides a consistent improvement over the individual parsing strategies.

LLMs like `Llama3-8B` and `CodeQWen` benefit the most from the MP+ approach, consistently achieving top scores across multiple categories. `Falcon` and `Pythia` demonstrate substantial improvements with MP+ over their GD and MP scores, suggesting that MP+ enhances both small-sized and large-sized LMs effectively.

Error analysis at the instance level Table 10 presents parsed texts generated by different LLMs using diverse parsing strategies for the question: *"Top 3 important features for id 3!"*. Tokens in the parsed text that are matched with the gold label are marked with underlines. None of the parsed texts match the gold label. Table 10 reveals that GD is good in generating top k values accurately, while MP and MP+ tent to correctly generate method names. However, there are instances where MP's generation is incomplete, e.g. the parsed text from `Pythia-2.8B` with MP lacking a numerical value for top k. Additionally, GD has a tendency to generate alternative method names like "lime" or "attention", when the "default" should be used when no method name is specified in the users' question (Table 6). Thus, Table 10 illustrates that when additional slots are available for operations, LLMs exhibit limitations in fully accurately recognizing every slot.

Name	Citation	Size	Link
`Falcon`	Penedo et al. (2023)	1B	https://huggingface.co/tiiuae/falcon-rw-1b
`Pythia`	Biderman et al. (2023)	2.8B	https://huggingface.co/EleutherAI/pythia-2.8b-v0
`Mistral`	Jiang et al. (2023)	7B	https://huggingface.co/mistralai/Mistral-7B-v0.1
`CodeQwen1.5`	Bai et al. (2023)	7B	https://huggingface.co/Qwen/CodeQwen1.5-7B-Chat
`sqlcoder`	n.a.*	7B	https://huggingface.co/defog/sqlcoder-7b-2
`Llama 3`	n.a.*	8B	https://huggingface.co/meta-llama/Meta-Llama-3-8B
`Llama 3`	n.a.*	70B	https://huggingface.co/meta-llama/Meta-Llama-3-70B

Table 8: Deployed LMs for parsing accuracy evaluation. *No paper published, with `GitHub` link only: https://github.com/meta-llama/llama3 and https://github.com/defog-ai/sqlcoder.

Category	Strat.	Falcon	Pythia	Mistral	Llama3-8B	Llama3-70B	CodeQWen	sqlcoder
Data	GD	63.43	89.77	71.88	91.67	85.42	77.08	80.21
Glb. Pr.	GD	72.97	93.14	93.14	100.00	100.00	83.33	100.00
Loc. Ex.	GD	53.85	80.77	80.77	84.62	84.62	73.08	84.62
Loc. Pr.	GD	66.67	100.00	100.00	100.00	100.00	66.67	100.00
Meta	GD	70.04	64.05	75.00	69.15	75.75	54.54	85.71
Modi.	GD	36.36	63.64	54.55	63.64	63.64	54.55	72.73
Pert.	GD	60.00	100.00	100.00	100.00	100.00	70.00	100.00
Data	MP	65.63	70.83	91.67	81.02	85.02	82.67	100.00
Glb. Pr.	MP	0.00	0.00	29.33	54.86	8.00	32.00	93.33
Loc. Ex.	MP	26.92	11.54	46.15	51.65	30.77	26.92	61.53
Loc. Pr.	MP	44.44	92.59	81.48	70.37	70.37	55.56	81.48
Meta	MP	85.02	88.89	79.05	67.70	96.77	76.94	80.56
Modi.	MP	63.63	81.82	90.91	81.82	72.73	90.91	81.82
Pert.	MP	100.00	100.00	100.00	60.00	70.00	90.00	50.00
Data	MP+	73.96	91.67	100.00	95.83	95.19	97.50	100.00
Glb. Pr.	MP+	69.45	68.14	80.55	86.77	91.11	84.55	89.63
Loc. Ex.	MP+	70.94	58.65	72.22	85.04	87.18	76.07	74.79
Loc. Pr.	MP+	44.44	100.00	81.48	70.37	100.00	66.67	88.89
Meta	MP+	87.40	88.89	82.94	72.78	93.23	78.71	82.24
Modi.	MP+	90.91	90.91	100.00	100.00	100.00	100.00	100.00
Pert.	MP+	100.00	90.00	100.00	100.00	100.00	100.00	90.00

Table 9: F_1 scores of each category for different LMs on CoXQL test set. GD = Guided Decoding prompted by 20-shots; MP = Multi-Prompt parsing; MP+ = MP with template checks.

Model	Strategy	Parsed Text	Correctness
Falcon-1B	GD	filter id 3 and nlpattribute topk 3 lime	✗
	MP	filter id 3 and nlpattribute attention all	✗
	MP+	filter id 3 and nlpattribute all default	✗
Pythia-2.8B	GD	filter id 3 and nlpattribute topk 3 lime	✗
	MP	filter id 3 and nlpattribute input_x_gradient topk	✗
	MP+	filter id 3 and nlpattribute all default	✗
Mistral-7B	GD	filter id 3 and nlpattribute topk 3 lime	✗
	MP	filter id 3 and nlpattribute all default	✗
	MP+	filter id 3 and nlpattribute all default	✗
CodeQwen1.5-7B	GD	filter id 3 and nlpattribute topk 3 attention	✗
	MP	filter id 3 and nlpattribute all default	✗
	MP+	filter id 3 and nlpattribute all default	✗
sqlcoder-7B	GD	filter id 3 and nlpattribute topk 3 lime	✗
	MP	filter id 3 and nlpattribute all default	✗
	MP+	filter id 3 and nlpattribute all default	✗
Llama3-8B	GD	filter id 3 and nlpattribute topk 3 attention	✗
	MP	filter id 3 and nlpattribute all default	✗
	MP+	filter id 3 and nlpattribute all default	✗
Llama3-70B	GD	filter id 3 and nlpattribute topk 3 attention	✗
	MP	filter id 3 and nlpattribute topk all	✗
	MP+	filter id 3 and nlpattribute all default	✗

Table 10: Parsed texts generated by various LMs employing different parsing strategies for the user question: "*Top 3 important features for id 3!*", where the gold label is `filter id 3 and nlpattribute topk 3 default`. Tokens associated with additional attributes that are matched with the gold label are marked with underlines. ✗ marks a parsed text that does not match the gold label. GD = Guided Decoding prompted by 20-shots; MP = Multi-prompt Parsing; MP+ = MP with template checks.