MetaExplainer In Action: An Overview of a Framework to Generate Multi-Type User-Centered Explanations*

Shruthi Chari[1], Oshani Seneviratne[1,*], Prithwish Chakraborty[2], Pablo Meyer[3] and Deborah L. McGuinness[1]

[1]*Rensselaer Polytechnic Institute, Troy, New York*
[2]*Amazon Science, New York*
[3]*IBM Research, Yorktown Heights, New York*

Abstract
Explanations are crucial for building trustworthy AI systems, but a gap often exists between the explanations provided by models and those needed by users. We build on prior research and direct feedback from clinicians, which reveal that users prefer interactive, question-driven, and diverse explanations. To bridge this gap, we present MetaExplainer, a neuro-symbolic framework that generates user-centered, multi-type explanations tailored to user questions. MetaExplainer follows a three-stage pipeline: (1) decompose user questions into machine-readable representations using state-of-the-art large language models (LLMs), (2) invoke appropriate model-specific explainer methods to generate recommendation rationales, and (3) synthesize coherent, user-friendly natural-language explanations that summarize and contextualize these outputs. We demonstrate MetaExplainer with an end-to-end example on the widely used PIMA Indian Diabetes dataset, highlighting how the framework addresses diverse clinical questions. Overall, MetaExplainer offers a versatile, traceable approach to explanation generation, capable of addressing a broad spectrum of user queries and advancing AI explainability across domains. MetaExplainer's implementation, along with quantitative and qualitative evaluation results, is detailed in [1], and the open-source code is publicly available at https://github.com/tetherless-world/metaexplainer.

Keywords
Explanation types, User-centered explainability, Neurosymbolic AI, Question-driven explanations

1. Introduction

Explainable AI has been a cornerstone of research for decades and has evolved in tandem with advancements in AI approaches. However, with the increase in complexity of AI methods, the results of explainer methods alone have been found insufficient for end-user needs. Several researchers [2, 3] have found that users require explanations for different purposes and as answers to various question types to effectively use explanations in their decision-making when interacting with AI systems. Further, several taxonomies [3, 4] have categorized explanation types, there remains a gap in generating real-time, user-centered explanations.

Hence, in the pursuit of supporting natural-language (NL) [5], diverse [6, 7] and multi-type user-centered [8] explainability, we introduce the MetaExplainer (Fig. 1), a multi-stage general-purpose explanation framework capable of generating explanations for end-user questions by summarizing explanations from several model explainers (such as SHAP [9], DiCE [10]) in NL. Specifically, with the MetaExplainer, we explore how interactions between various sub-components of explanation systems, i.e., AI reasoning, prompt-driven interactions with large language models (LLMs) enriched with knowledge from the Explanation Ontology (EO) [11] and explainer methods, can help generate explanations of AI systems. In this paper, we provide an overview of the framework with an end-to-end example of how the MetaExplainer generates NL explanations.

ISWC 2025 Companion Volume, November 2–6, 2025, Nara, Japan
*Corresponding author.
charishruthi@gmail.com (S. Chari); senevo@rpi.edu (O. Seneviratne)
0000-0003-2946-7870 (S. Chari); 0000-0001-8518-917X (O. Seneviratne); 0000-0003-1407-7677 (P. Chakraborty); 0000-0001-5281-4661 (P. Meyer); 0000-0001-7037-4567 (D. L. McGuinness)

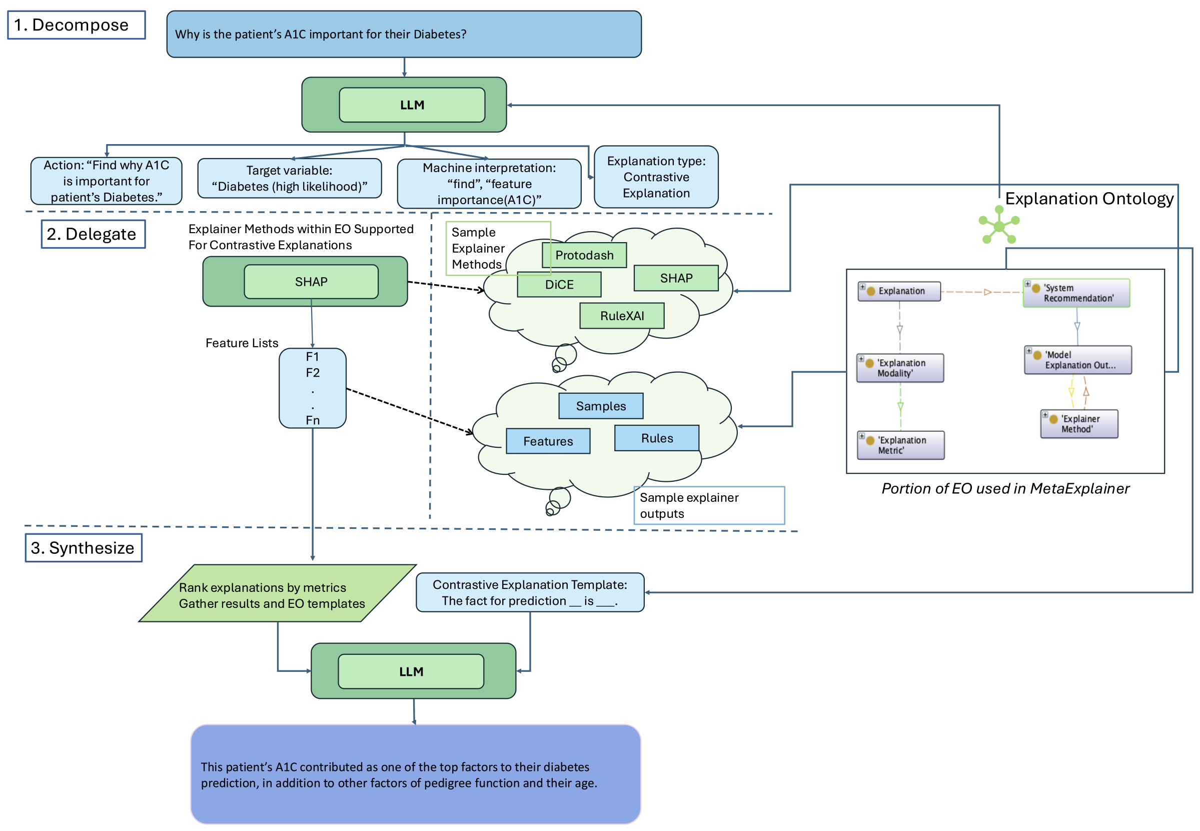
1. Decompose
Why is the patient's A1C important for their Diabetes?
LLM
Action: "Find why A1C is important for patient's Diabetes."
Target variable: "Diabetes (high likelihood)"
Machine interpretation: "find", "feature importance(A1C)"
Explanation type: Contrastive Explanation
Explanation Ontology
2. Delegate
Explainer Methods within EO Supported For Contrastive Explanations
SHAP
Sample Explainer Methods
Protodash
DiCE
SHAP
RuleXAI
Feature Lists
F1
F2
Fn
Samples
Features
Rules
Sample explainer outputs
Explanation
'System Recommendation'
'Explanation Modality'
'Model Explanation Out...
'Explainer Method'
'Explanation Metric'
Portion of EO used in MetaExplainer
3. Synthesize
Rank explanations by metrics Gather results and EO templates
Contrastive Explanation Template: The fact for prediction __ is ___.
LLM
This patient's A1C contributed as one of the top factors to their diabetes prediction, in addition to other factors of pedigree function and their age.

Figure 1: Workflow diagram of MetaExplainer, highlighting the different input and output streams. Methods are indicated by green boxes and all other data components are shown in blue.

2. Method Overview

Our MetaExplainer is a framework that can generate natural-language explanations in response to user questions, along explanation types (Tab. 1) we currently support. In the design of the MetaExplainer (Fig. 1), we adopt a modular approach - in that we break down the design into three stages; *decompose:* convert the user question into a machine interpretation that can then be *delegated:* to explainer methods registered against explanation types in EO and finally, the explainer outputs are *synthesized* into NL explanations presented to the end-users. We implement the framework as a Python package, such that each of the stages can call methods and classes from one another , enabling the stages to build off of the intermediary stage outputs as seen in Fig. 1.

Decompose: In the Decompose stage, our objective is to generate a machine-actionable parse of a user question(s), including the explanation type that best addresses the question and the features being asked about. For example, in a question - "Why is a 60-year-old woman with a BMI of 28 more likely to have Diabetes?", we would want to identify that this question is best addressed by facts in contrastive explanations and has filters applied on features - age, BMI. These attributes, if captured, help ensure that the explainer methods can reliably address the user's question. We find that predicate logic can serve as a good intermediate to parse in further stages of the MetaExplainer. Hence, we build a question bank of *uq*: *rq* pairs (leveraging question type-explanation type mappings in EO) for each explanation type, expand this question bank using GPT, and fine-tune LLama2 and Llama3 LLMs on this question bank to generate *rq* from *uq*. Hence, in the Decompose stage, we set up a task to convert a user question *uq* into a reframed question *rq*, which is a tuple capturing the {question, explanation type, machine interpretation, action}.

Delegate: In the Delegate stage, the objective is to execute relevant explainer methods to address a user question (*uq*). Here, we further break it down into sub-tasks: parsing the *rq*, executing explainers

(*em*), and processing explainer outputs (*eq*). In the parsing task, we parse the machine interpretation field of the *rq* for filters on features, the explanation type (*et*) field to identify what explainer methods (*em*) can be run, and the action field to identify if there are other actions the question asks about (e.g., preprocessing, accuracies, etc.) In the execution task, we run open-source explainer methods (*em*) (Tab. 1), e.g. SHAP for contrastive explanations. We leverage the explanation type - explainer method mappings in the EO to identify what explainers to run. Finally, in the output processing step, we persist the explainer outputs as dataframes based on the explanation modality (i.e., features, instances or rules, again mappings captured in the EO), that the Synthesis stage can then use.
Synthesis: In the Synthesis stage, we combine and synthesize the explainer output(s) into natural-language (NL) explanations aligned with the identified explanation type's (*et*) expected templates. We further break down the Synthesis into three sub-tasks, including a retrieval step where we extract relevant data points from the explainer outputs to include in the final explanation, an augmentation/alignment step where we align the outputs to fit the identified explanation type's (*et*) templates, and a generation step where we output NL explanation ($\mathscr{E}_q$) populated with the retrieved content in line with the template for the *et*. Hence, the synthesis task lends itself well to an application of the widely popular Retrieval-Augmented-Generation (RAG) [12], [13] technique, wherein we design prompts enriched with *et* to generate NL explanations (*E*) by retrieving relevant content from explainer outputs (*eq*) from the previous stage. We generate two explanations for each question, one that summarizes the explainer outputs and the other that summarizes the subset of records that matched the filters in user question, *uq*.

Algorithm 1 Pseudocode for MetaExplainer indicating the three stages - Decompose, Delegate, and Synthesis, and their inputs and outputs.

```
Require: Explanation Ontology (EO), Data Store (DS)

    Explanation type - explainer graph, $G^1 = \{(t_i, em_j), \forall i \in N \text{ and } \forall j \in M\}$

    Data type - explainer graph, $G^2 = \{(d_k, em_j), \forall k \in K \text{ and } \forall j \in M\}$

    Input
        $uq$            User Question

    $rq$ = Decompose($uq$)
    $eq$ = Delegate($rq$)
    $E$ = Synthesis($eq$)

    Output
        $rq$            list of questions reframed from $uq$
        $E$             list of explanations $\{E_e\}$ that answer $uq$ where $E_e = \{\mathscr{E}_e, t_e, em_e, rq_e, uq_e\}$
```

2.1. Explanation Types supported in the MetaExplainer

We currently support five explanation types in the MetaExplainer and explainer methods that can generate explanations along the same, as shown in Tab. 1. System developers could add support for more explanation types by including the explainer methods in the Delegate stage of the MetaExplaier and registering the expected outputs and question types in the EO.

3. MetaExplainer in Action: Diabetes Use Case

We evaluated the MetaExplainer, by applying the five explainer methods (Tab. 1) we currently support in the Delegate stage on the best performing predictive model, a random forest (RF) model on the

Table 1
Explanation types and explainer methods we currently support in the MetaExplainer

Explanation Type	Definition	Explainer Method
Case-based	Provides solutions that are based on actual prior cases that can be presented to the user to provide compelling support for the system's conclusions and may involve analogical reasoning, relying on similarities between features of the case and of the current situation.	Protodash
Contrastive	Answers the question "Why this output instead of that output," making a contrast between the given output and the facts that led to it (inputs and other considerations), and an alternate output of interest and the foil (facts that would have led to it).	SHAP
Counterfactual	Addresses the question of what solutions would have been obtained with a different set of inputs than those used.	DiCE
Data	Focuses on what the data is and how it has been used in a particular decision, as well as what data and how it has been used to train and test the ML model. This type of explanation can help users understand the influence of data on decisions.	Protodash
Rationale	About the "why" of an ML decision, and provides reasons that led to a decision, and is delivered in an accessible and understandable way, especially for lay users.	RuleXAI

widely used tabular dataset, PIMA Indian Diabetes [14] dataset. We generated a question bank of user questions (*uq*) around the features in predicting whether patients have Diabetes or not, and use these to design and test the MetaExplainer. Our results show good performance across all stages of the MetaExplainer, with a 59.06% F1-score in question reframing in the Decompose stage, 70% faithfulness in model explanations in the Delegate stage, and 67% context-utilization in NL synthesis. We describe our results in our full-paper linked in Sec. 1.

3.1. End-to-end example of user question being addressed by the MetaExplainer

```
Question: How did the model justify predicting Diabetes for a 55-year-old male with a BMI of 18 and a
     Diabetes Pedigree Function of 0.25?

Explanation Type: Rationale Explanation

Machine Interpretation:
    Predict(Diabetes, Age = 55, Sex = Male, BMI = 18, DPF = 0.25)

RuleXAI output:
    1. IF BMI = (<32.25, inf) THEN label = {1}
    2. IF BMI (-inf, 37.05) THEN label = {1}

Explainer Method Metrics:
    Average rule length - 1.4
    Fidelity: 0.4
```

Figure 2: Decompose and Delegate outputs from the MetaExplainer for an example question in a Diabetes prediction use case

As shown in Fig.s 2 and 3, our MetaExplainer framework processes a user question, *uq*: "How did the model justify predicting diabetes for a 55-year-old male with a BMI of 18 and a diabetes pedigree function of 0.25?"

In Fig. 2, the user question, *uq*, is translated into its predicate logic equivalent, *rq*, and a rationale explanation type is identified during the *Decompose* stage. The fields for machine interpretation and explanation type are then passed to the *Delegate* stage. Here, the RuleXAI post-hoc explainer method [15] generates rationales or trace-based explanations, outputting rules that might have been used in the classification model. Further, we quantify the rules using relevant explanation metrics for the rule explanation modality [16]—specifically, rule length and fidelity metrics for rule-based explanations. The generated rules have an average rule length of 1.4 and a fidelity of 0.4.

Finally, as depicted in Fig. 3, the rules and the user question, uq, are utilized by the Synthesis stage to produce NL summaries, E_q, of the subset of data that matched the feature group restrictions in the question and the rules responsible for the classification model's prediction of diabetes for a 55-year-old male with a BMI of 18 and a diabetes pedigree function of 0.25. This illustrates how each stage of the MetaExplainer builds upon the outputs of previous stages. Our modular, multi-stage framework effectively captures the restrictions in the question and traces back outputs that a purely LLM approach might miss.

```
Explanation of Matched subset:
    There are no full matches in the dataset based on the specified feature group. However, the dataset
         has been summarized, showing the descriptive statistics for the variables in the dataset.
    - The mean age is 68.1 years.
    - The mean BMI is 29.49.
    - The mean Diabetes Pedigree Function (DPF) is 0.4868.
    - The outcome variable has a mean value of 0.3, indicating that the majority of the observations
         have an outcome of 0.

Explanation of explainer output:
    The rules derived for classification based on the features
    'Age' and 'BMI' are as follows:
    1. IF BMI is less than or equal to 32.25, THEN label = 0
    2. IF BMI is greater than 37.05, THEN label = 1

These rules provide a clear guideline for classifying data based on BMI values, with different labels
     assigned depending on the BMI range. This type of explanation helps users understand the rationale
     behind the decision-making process in machine learning models, allowing them to assess the
     reasoning and make informed judgments.
```

Figure 3: Natural-language Explanations of Diabetes Prediction from Synthesis stage of MetaExplainer

4. Conclusion

We provide an overview of the MetaExplainer, a multi-stage explanation generation framework that we have designed and implemented, that combines several AI and machine learning methods - LLMs, ontologies, predictive and explainer models- to generate user-centered explanations. We evaluate our framework both qualitatively and quantiatively (evaluating outputs at each stage) on an indicative healthcare use case, Diabetes, with plans and provisions to expand beyond. Since our framework is modular, system developers can add support for explanation types beyond those currently supported. We have open-sourced our implementation and invite the community to contribute. Overall, the MetaExplainer helps address a timely problem of tailoring explanations to end-user needs and leverages the strengths of a neuro-symbolic approach to do so.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to rephrase some of the sentences and also to fix grammar and spelling issues. After using these tools and services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Chari, O. Seneviratne, P. Chakraborty, P. Meyer, D. L. McGuinness, Metaexplainer: A framework to generate multi-type user-centered explanations for ai systems, arXiv preprint arXiv:2508.00300 (2025).

[2] S. Tonekaboni, S. Joshi, M. D. McCradden, A. Goldenberg, What clinicians want: contextualizing explainable machine learning for clinical end use, in: Mach. Learn. for Health. Conf. (ML4HC), 2019, pp. 359–380.

[3] D. Wang, Q. Yang, A. Abdul, B. Y. Lim, Designing theory-driven user-centric explainable ai, in: Proc. of the 2019 Conf. on Human Factors in Comp. Syst. (CHI), 2019, pp. 1–15.

[4] Q. V. Liao, D. Gruen, S. Miller, Questioning the ai: informing design practices for explainable ai user experiences, in: Proc. of the 2020 Conf. on Human Factors in Comp. Syst. (CHI), 2020, pp. 1–15.

[5] H. Lakkaraju, D. Slack, Y. Chen, C. Tan, S. Singh, Rethinking explainability as a dialogue: A practitioner's perspective, 2022. ArXiv2202.01875.

[6] B. Mittelstadt, C. Russell, S. Wachter, Explaining explanations in ai, in: Proc. of the Conf. on Fairness, Accountability, and Transparency, 2019, pp. 279–288.

[7] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artif. Intell. 267 (Feb. 2019) 1–38.

[8] S. Dey, P. Chakraborty, B. C. Kwon, A. Dhurandhar, M. Ghalwash, F. J. S. Saiz, K. Ng, D. Sow, K. R. Varshney, P. Meyer, Human-centered explainability for life sciences, healthcare, and medical informatics, Patterns 3 (May 2022).

[9] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Infor. Process. Syst., 2017, pp. 4768 – 4777.

[10] R. K. Mothilal, A. Sharma, C. Tan, Explaining machine learning classifiers through diverse counterfactual explanations, in: Proc. of the Conf. on Fairness, Accountability, and Transparency, 2020, pp. 607–617.

[11] S. Chari, O. Seneviratne, M. Ghalwash, S. Shirai, D. M. Gruen, P. Meyer, P. Chakraborty, D. L. McGuinness, Explanation ontology: A general-purpose, semantic representation for supporting user-centered explanations, Semantic Web J. Pre-press (May 2023) 1–31.

[12] 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, 2023. ArXiv2312.10997.

[13] 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, in: Advances in Neural Inf. Process. Syst. (NeurIPS), 2020, pp. 9459–9474.

[14] V. Chang, J. Bailey, Q. A. Xu, Z. Sun, Pima indians diabetes mellitus classification based on machine learning (ml) algorithms, Neural Compu. and Appl. 35 (Mar. 2023).

[15] D. Macha, M. Kozielski, Ł. Wróbel, M. Sikora, Rulexai—a package for rule-based explanations of machine learning model, SoftwareX 20 (Dec. 2022).

[16] J. van der Waa, E. Nieuwburg, A. Cremers, M. Neerincx, Evaluating xai: A comparison of rule-based and example-based explanations, Artif. Intell. 291 (2021).