RUBICON: Rubric-Based Evaluation of Domain-Specific Human-AI Conversations

Param Biyani*
t-pbiyani@microsoft.com
Microsoft, India

Yasharth Bajpai*†
ybajpai@microsoft.com
Microsoft, India

Arjun Radhakrishna
arradha@microsoft.com
Microsoft, Redmond

Gustavo Soares
gustavo.soares@microsoft.com
Microsoft, Redmond

Sumit Gulwani
sumitg@microsoft.com
Microsoft, Redmond

ABSTRACT

The evaluation of conversational assistants, such as GitHub Copilot Chat, poses a significant challenge for tool builders in the domain of Software Engineering. These assistants rely on language models and chat-based user experiences, making evaluating them according to the quality of the Human-AI conversations complicated. Existing general-purpose conversational quality metrics from literature are inadequate for assessing domain-specific dialogues due to their lack of context sensitivity. In this paper, we present RUBICON, a technique for evaluating domain-specific Human-AI conversations. RUBICON leverages large language models to generate rubrics for assessing conversation quality. It employs a selection process to choose the subset of rubrics based on their performance in scoring conversations. In our experiments, RUBICON effectively learns to differentiate conversation quality, achieving higher accuracy and yield rates than existing baselines.

ACM Reference Format:
Param Biyani, Yasharth Bajpai, Arjun Radhakrishna, Gustavo Soares, and Sumit Gulwani. 2024. RUBICON: Rubric-Based Evaluation of Domain-Specific Human-AI Conversations. In *Proceedings of ACM Conference (Manuscript Submitted to ACM).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

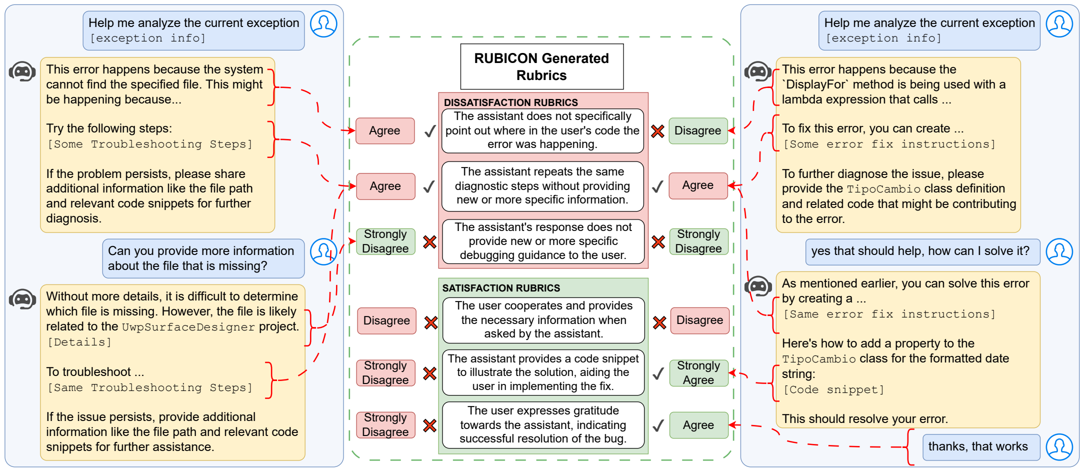
With the introduction of ChatGPT [1], AI assistants for software developers, such as Github Copilot [11], have started to adopt chat-based interfaces as one of the primary means for enabling interactive AI assistance for developers. However, evaluating the quality of these chat-based AI assistants has proven to be highly challenging due to the diverse range of tasks developers seek help with and the multi-turn nature of Human-AI conversations, making it difficult to determine the success of the interaction with the assistant. As a result, developers who build applications with chat interfaces often lack sufficient information regarding the performance and quality of the interactions facilitated by their tools [22].

Recently, Lin et al. [16] introduced SPUR, a technique designed to estimate User Satisfaction for open-domain conversations conducted with Bing Copilot. First, SPUR leverages a large language model to analyze past labelled conversations, capturing user satisfaction signals from upvoted conversations and user dissatisfaction signals from downvoted conversations. These signals are then utilized to iteratively generate a set of rubrics. Next, SPUR empowers a large language model to assess the quality of new conversations by scoring them against the learned set of rubrics. While user satisfaction rubrics learned by SPUR can be effective in assessing Human-AI conversations that contain user prompts indicating satisfaction or dissatisfaction, such as compliments or rephrased questions, they may be less effective in conversations where such signals are absent.

For example, consider Figure 1 where we have two similar conversations evaluated against some rubrics generated from cues from the conversation. The left conversation is an example of the user being unable to get the required help from the assistant as the assistant does not have access to certain information it asked for in the previous step. While in the conversation on the right, the user gets the assistant to get code-fix suggestions on their further clarification about help in solving the issue which works for them. The rubrics are generated from relevant conversational cues to gauge the quality of conversation. These cues encompass repetition of information, progress towards task, adherence to user requests, provision of requisite information, etc. Domain-specific rubrics like ‘provides a code snippet as solution’ and ‘does not provide new or specific debugging guidance’ appear to be more contextual and definitive in capturing the essence of the conversation compared to generic counterparts like ‘provides a solution’ and ‘helps the user’ available from SPUR [16]. Nevertheless, the ability of large language models to identify signals and generate rubrics to assess conversations has inspired further exploration of different types of signals beyond user satisfaction, particularly in task-oriented settings where notions of satisfaction are tied to the task’s specific needs.

In this paper, we propose RUBICON, a technique for evaluating domain-specific Human-AI conversations. RUBICON comprises three main components: (1) rubric set generation, (2) rubric selection, and (3) conversation evaluation. In the first step, RUBICON follows the approach of SPUR to identify signals in conversations labelled as *negative* and *positive*, generating a set of rubrics. However, RUBICON extends SPUR by providing instructions to the model regarding domain-specific signals (DS) and conversation design

*Equal Contribution; more junior author listed earlier
†Corresponding Author


Help me analyze the current exception
`[exception info]`

This error happens because the system cannot find the specified file. This might be happening because...

Try the following steps:
`[Some Troubleshooting Steps]`

If the problem persists, please share additional information like the file path and relevant code snippets for further diagnosis.

Can you provide more information about the file that is missing?

Without more details, it is difficult to determine which file is missing. However, the file is likely related to the `UwpSurfaceDesigner` project.
`[Details]`

To troubleshoot ...
`[Same Troubleshooting Steps]`

If the issue persists, provide additional information like the file path and relevant code snippets for further assistance.

RUBICON Generated Rubrics

DISSATISFACTION RUBRICS

The assistant does not specifically point out where in the user's code the error was happening.

The assistant repeats the same diagnostic steps without providing new or more specific information.

The assistant's response does not provide new or more specific debugging guidance to the user.

SATISFACTION RUBRICS

The user cooperates and provides the necessary information when asked by the assistant.

The assistant provides a code snippet to illustrate the solution, aiding the user in implementing the fix.

The user expresses gratitude towards the assistant, indicating successful resolution of the bug.

Help me analyze the current exception
`[exception info]`

This error happens because the `DisplayFor` method is being used with a lambda expression that calls ...

To fix this error, you can create ...
`[Some error fix instructions]`

To further diagnose the issue, please provide the `TipoCambio` class definition and related code that might be contributing to the error.

yes that should help, how can I solve it?

As mentioned earlier, you can solve this error by creating a ...
`[Same error fix instructions]`

Here's how to add a property to the `TipoCambio` class for the formatted date string:
`[Code snippet]`

This should resolve your error.

thanks, that works

Figure 1: Two analogous conversations facilitated by the Debugger AI assistant evaluated against some representative rubrics. The conversation on the right was deemed better, while the one on the left was considered subpar.

principles (CDP) in the form *Gricean maxims* [30], which capture four dimensions of conversation effectiveness: quantity, quality, relevance, and manner. Additionally, while SPUR combines the rubrics produced in each batch of examples without evaluating the effectiveness of the final rubric set, RUBICON generates fresh rubrics for each batch to create a pool. In the second step, RUBICON, inspired by prompt optimization using multi-arm bandit selection in Pryzant et al. [25], iteratively selects rubrics based on two data-driven loss functions. This step yields the final set of rubrics and a score threshold for classifying conversations as negative or positive. Finally, in the last step, RUBICON employs a large language model to grade and classify the conversation under test using the generated rubric set and threshold.

For our evaluation, we instantiated RUBICON to assess conversations between developers and a widely used chat-based assistant designed for C# developers. To create a labelled dataset, we collected 100 conversations where developers sought assistance from the chat-based assistant to resolve an exception while debugging their code. To power RUBICON, we utilized GPT-4[20], a state-of-the-art language model. GPT-4 was employed to generate rubrics, select the most relevant ones, and assess the conversations based on the learned rubric set and scoring threshold.

The rubrics generated using RUBICON exhibited superior performance, creating a 9x higher delta in the score of negative and positive conversations, compared to three alternative sets of rubrics: the original set from [16], a manually adapted set, and a set learned from our debugging dataset using SPUR. We are able to predict conversation labels with an almost perfect (> 0.9) precision for 84% of conversations on unlabelled data. Additionally, we conducted ablation studies to show the effect of each component of our technique.

Overall, we present the following contributions:

- We propose RUBICON a technique to automatically generate rubrics and score conversations for AI Assistants for specific domains;
- We compare the rubric performance when generated by RUBICON and SPUR, using a dataset of 100 real conversations between developers and an AI assistant in the context of debugging exceptions;
- We evaluate the impact and effectiveness of the different components within RUBICON.

2 RELATED WORK

Natural language conversations have become the staple interface for modern AI applications [8, 13, 29]. Traditional NLP techniques have employed many metrics, but in the LLM age, not many have matured in comparison to the advancement in the use cases [7, 15, 21]. Metrics like BLEU [21], RoBERTA [17], and Perplexity are often found short on being able to measure long-form conversations, particularly due to lack of ideal references and underlying intent. [18, 26, 34, 36].

In the broader sense, conversation quality analysis is not a new problem, and domains, particularly sales and marketing, have employed various ways of measuring the quality of conversation their end-users experience when interacting with agents [3]. User satisfaction estimation has been used as a proxy for evaluating conversational quality from a user experience-led approach [16]. User satisfaction through surveys and manual post-analysis of the conversation has been the most important of all other indicators. However, manual post-analysis of conversations has both privacy and resource implications, thus difficult to carry out at scale [10, 14, 32]. Human annotations of conversations themselves are subjective and are known to be prone to bias [6, 9, 12, 19].

Given that data annotation and large-scale user study is a tedious and resource consuming task. Recent works have explored using language models to replace human subjects in answering evaluation-related questions. These works propose scoring systems built around judging conversations against some assertions [10, 16, 19, 27]. Assertions are natural language sentences framed to ask specific questions about any conversation. These assertions can be designed to detect the presence of themes ranging from user experience - frustration, curiosity, satisfaction, impatience, etc., to more sophisticated/abstract themes like engagement, inconsistency, interestingness, understanding, etc.

This paper also discusses SPUR's [16] proposed automatic technique, which uses general open domain conversations with thumbs up/down user response signals to generate assertions for user satisfaction estimation. However, this approach and other domain-agnostic user satisfaction estimation policies like [35] are only designed for open-ended conversations. The assertions generated in SPUR are designed to measure only the signals of user satisfaction and serve as its proxy. However, for a complete conversational quality evaluation, the literature in HCI and other relevant fields focusing on Human-AI conversations advocate for a holistic approach to designing conversational AI systems, emphasizing factors such as naturalness, engagement, trust, empathy, and context awareness to create meaningful interactions that go beyond mere user satisfaction. Cathy [23] and Semnick [28] emphasize the importance of understanding user needs, expectations, and conversational norms to create engaging and satisfying experiences. A more well-rounded conversational evaluation, especially for task-oriented interactions, would account for understanding expectations and the progress the interaction enables in that direction [4, 7].

Evaluating multi-turn conversations typically uses scoring over a Likert scale after a conversation is over [5, 10, 27, 32, 37]. Some works also explore using continuous scales to rate conversations per assertion [16]. However, we find that LLMs are usually inconsistent and biased when giving numeric ratings to natural language assertions. In this paper, we briefly discuss this trade-off and other than for comparison purposes, we only use the Likert scale throughout.

In this work, we also explore the problem of selecting an optimal subset of prompts from a larger set of prompts and compare our approach with some bandit selection methods [25]. The selection policy is optimised for two metrics related to binary classification: distance between the means of the distribution of the two classes, and the percentage of the conversations that can be evaluated with some minimum confidence threshold. Since it is challenging to perform binary classification on dialogues due to their subjectivity, we only consider conversations classified with high confidence after the scoring. Similar techniques have been applied in problems like credit default prediction or fraud detection where the precision requirements of classification are very strict [31, 33].

3 TECHNIQUE

We propose RUBICON to estimate conversation quality for domain-specific conversational assistants. To enable this, we learn rubrics capturing Satisfaction (SAT) & Dissatisfaction (DSAT) for a given conversation set with binary ground labels - positive & negative; which are then used to obtain a score $NetSAT$ in an online fashion while classifying them on a learned threshold θ. Fig 2 outlines three major components in our technique - (1) Generating a diverse rubric set from conversation data; (2) Selecting an optimized set of rubrics for online evaluation; (3) Scoring conversations and predicting labels. We refer to (1) and (2) as the Generate & Select paradigm of learning rubrics which are carried out in an offline setting.

The Generation step is inspired from SPUR to support learning of attributes and patterns of Satisfaction/Dissatisfaction from the available trainable conversations C_{train}. We adapt *Supervised Extraction* proposed in SPUR to include domain sensitization as well as conversation design insights, while the manipulated *Rubric Summarization* adheres to standards of conversation while accumulating the set of diverse assertions. Secondly, our novel selection policy optimizes the number of assertions to help converge to rubric subsets of restricted sizes making online evaluation feasible. The policy incorporates components to address concerns in a data-scarce environment while providing a better opportunity for labelling with high precision and coverage. Each rubric is a natural language assertion evaluated against a 5-segment Likert scale using an instruction-tuned LLM. SAT/DSAT rubrics in the optimized set R_S with a fixed N rubrics of each undergo assessment in the wild to produce $NetSAT$ score, with the premise that an ideally positive conversation should exhibit a higher overall SAT score and a lower DSAT score.

3.1 Rubrics and Conversation Quality

The core component of our system is a *rubric* r, i.e., a natural language assertion of some property of a human-AI assistant conversation. For example, "the user explicitly thanks the assistant" and "the information provided by the AI assistant was generic, and did not consider the user's current problem" are both rubrics. We classify rubrics into *satisfaction rubrics* and *dissatisfaction rubrics* (denoted SAT and DSAT) to say whether the rubric expresses a positive or a negative sentiment about the conversation. We use the symbols s and d to represent SAT and DSAT rubrics respectively.

Given a conversation c, we can *evaluate* c with respect to a rubric r by providing both of them to a language model and asking it to rate the match on a 5-point Likert scale. This 5 point score ranging from *Strongly Disagree* to *Strongly Agree* is converted into a normalized score in the $[0, 10]$ range. For ease of discussion, the score is negated (i.e., in the range $[-10, 0]$) for DSAT rubrics to ensure that a larger score is always better. We denote this normalized score as $\mathrm{eval}(r, c)$.

Rather than using a generic prompt for evaluation, we provide domain & task sensitized instructions regarding the evaluation of a rubric with respect to a conversation with an instruction-tuned LLM. Our experiments (ref 4.5.4) show that these changes help the model score the rubrics in a more aligned way. Given input size limitations for the LLM and varying sizes of conversation to be scored, the evaluations are carried out in batches of ≤ 10 rubrics at a time with SAT & DSAT scored separately.

For a set of Rubrics R_S comprising of SAT and DSAT rubrics (N each), we define the conversation score, which we refer to as $NetSAT$. Similar to [16], the score measures the net satisfaction by subtracting (magnitude) the total from the DSAT rubrics from the total from the SAT rubrics.

$$NetSAT(R_S, c) = \sum_{s \in R_S} \mathrm{eval}(s, c) + \sum_{d \in R_S} \mathrm{eval}(d, c)$$

3.2 Generation of Rubrics

The first component of the technique, as shown in Figure 2, is to generate a rubric pool, given a training dataset of conversations C partitioned into positive and negative conversations C_+ and C_-. Inspired by SPUR [16], we use a two phase strategy. First, we perform a *supervised extraction* of conversation quality reasonings and patterns. Then, we summarize these reasonings in batches and

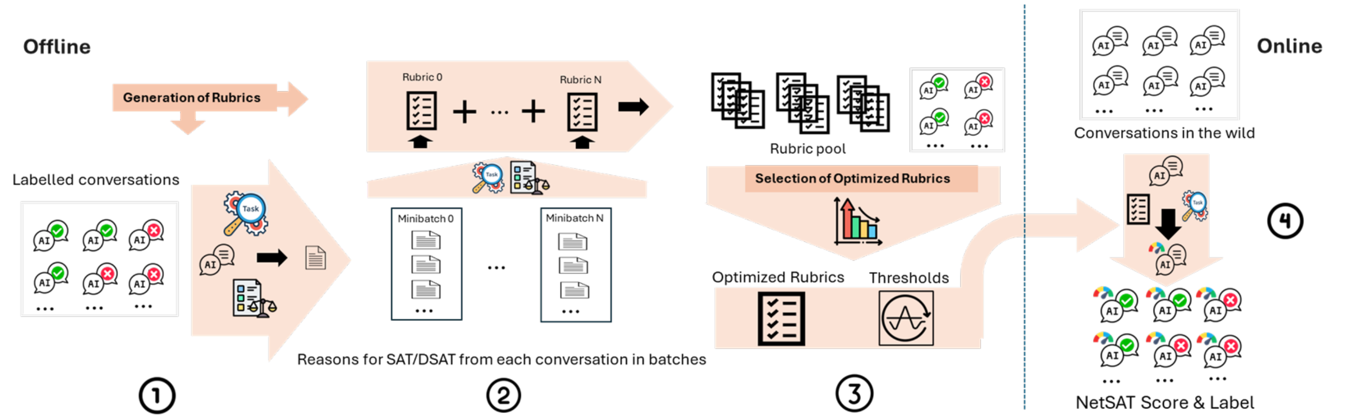


Figure 2: Representative Overview of the RUBICON.
(1) Extract task-specific patterns aligned with conversation design principles from each conversation. (2) Generate a large rubric pool from extracted reasons in the context of Gricean Maxims & the task. (3) Use data-driven loss to select optimal rubrics over conversation data for online evaluation (4) Score the quality of each conversation over the learnt rubrics and thresholds

put them into a large rubric pool. Both the supervised extraction step and the rubric summarization steps are modified significantly from the versions in [16], and we compare them experimentally and report the results in Section 4.

3.2.1 Supervised Extraction. Given a conversation $c \in C_{\text{train}}$, the supervised extraction step attempts to capture patterns $\hat{r}_i$ for particular satisfaction and dissatisfaction aspects of the conversation. In the conversation is annotated as positive in the ground-truth training set, we extract satisfaction patterns $\hat{s}_i$ and dissatisfaction patterns $\hat{d}_i$ otherwise. Like in [16], we extract the top-k patterns from each conversation ($k = 3$ in the experiments).

Example 3.1. Sample Reasonings for conversations from Fig. 1. *When the user asked for further clarification, the assistant provided a comprehensive and actionable plan in form of a detailed step-by-step guide on how to implement the solution, including code examples.* (SAT, right conversation)
The assistant was unable to provide more specific information about the missing file, which was the user's direct question, due to lack of additional information from the user. (DSAT, left conversation)

As evident in 3.1, these reasonings are picked from localized patterns in the exchanges that the user had an understanding of why they might be satisfied/dissatisfied with this conversation.

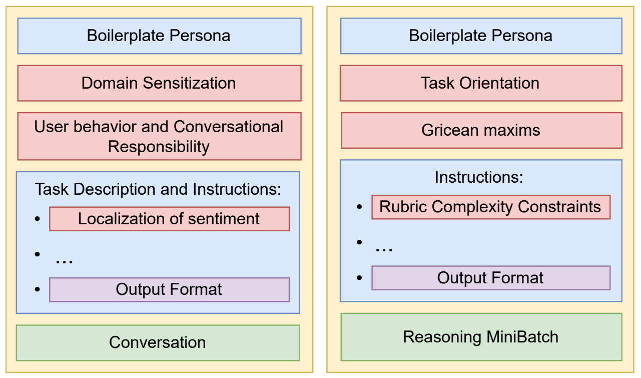
The outline of the prompt we use for supervised extraction is presented in Figure 3(a). Below, we discuss the salient, novel aspects of our supervised extraction.

User behaviour and Conversational Responsibility(CDP). The field of classical conversation analysis assigns responsibility for conversation quality to both parties involved [23, 24]. In a conversational AI assistant setting, it is easy to ignore the responsibility of the user as a first-class participant in the conversation, and assign the full responsibility of driving a positive conversation towards completing the task to the assistant and in turn, the tool builder. In fact, prior works have done so [19, 27]. However, the prompt shown in Figure 3, presented a dedicated block for us to add conversation responsibility. In our study, we explicitly ask the model to take into account how both the user and the assistant took steps to either progress or hinder the conversation towards task completion. Fig 1 (left) shows a redacted conversation where the user fails to provide the relevant information the assistant asks for.

Reasonings may point out issues with the user experience of the tool and in the user's understanding of its capabilities. For example, based on the following user-related reasoning - *"The assistant was unable to access the file linked by the user, which hindered the progress of the conversation and likely caused frustration for the user."*, the builders of the debugging assistant we test in our experiments are able to identify two shortcomings in the design of the tool: (a) there were insufficient UX affordances for users to provide new files to the assistant, and (b) The interface design sometimes lead to users confusing the debugging assistant with other IDE assistants and trying to ask non-debugging related questions of the assistant.

Domain Sensitization and Task Orientation (DS). We aim to produce rubrics that measure the quality of a conversation that is specifically about achieving a domain-specific goal (for example, debugging to fix an exception). That is, it does not matter if the assistant and user have an engaging conversation about the weather in Berlin if it is not goal-directed. Hence, it is important that the prompt to extract reasonings specifically mentions aspects of what the task is, expectations, and desired goals of the interaction. For example, in the debugging domain, our prompt contains specific describing expectations from the assistant as *"designed to hold a conversation to understand, ask for more information, and investigate the bug, then provide solutions to aid the user in their debugging tasks."* This matches the intuition that for a task-oriented Human-Human conversation to be positive, it needs to both have good "conversational" qualities (e.g., participants know their roles, neither participant is frustrated, or dominating the conversation, etc) and task-specific qualities (e.g., progress made towards fixing the bug, etc).

3.2.2 Rubric Summarization. Here, we consolidate the patterns generated in the Supervised Extraction step into SAT and DSAT



(a) Supervised Extraction **(b) Rubric Summarization**

Figure 3: Rubric Generation Prompt Outlines.

rubrics. Unlike in [16], we do not aim to produce a small and usable set of rubrics updated at each step, but instead to produce a large pool of diverse rubrics. The next step will select a smaller optimized subset of good quality rubrics from this pool.

This step incrementally generates new rubrics by processing a mini-batch of patterns identified in the step. Let the i^{th} minibatch be denoted as $\{\hat{r}_{m*(i-1)}, ..., \hat{r}_{m*i}\}$. For every minibatch, we prompt the LLM to generate a rubric set R_i based on its reasoning minibatch while also providing rubrics generated until the previous minibatch $\{R_1, ..R_{i-1}\}$. The goal is to have the generated rubric set R_i cover all the patterns in the minibatch *while* diversify over and beyond the previously generated rubrics. The SAT and DSAT patterns are batched separately, with corresponding changes in the prompts to generate either SAT or DSAT rubrics respectively. An important point of difference with SPUR is that we *do not update* a restricted set of N current rubrics R_i at each turn and instead augment the rubric pool with fresh rubrics at each batch. A restricted set is then selected from a larger pool in a data-driven manner, instead of asking the LLM to update the rubrics based on reasonings in each batch.

After obtaining the pool of rubrics R consisting of all SAT/DSAT rubrics, we conduct a post-processing step to *semantically de-duplicate* the set removing rubrics that exhibit semantic similarity to others. This is done to avoid any redundancy and bias in the final rubric set after selection. If r and r' are two well performing semantically equivalent rubrics in the pool, they might likely end up in the final rubric set after selection (ref 3.3), which may overpower and skew the resultant scores for the conversations using this set. To execute this, we use the reasoning capabilities of GPT4 with a curated instruction prompt (details in [2]) to remove any duplicate rubrics that cover similar reasonings, ideas or questions.

Gricean Maxims (CDP). We provide Grice's maxims as a framework of a cooperative principle for an effective conversation. These rational principles for improving conversation quality are encapsulated within four maxims: quantity, quality, relevance, and manner. These enable the summarizer to gauge different aspects of the conversation while generating the rubrics, particularly aspects like lapse on the part of any user, digression, confusion and lack of a proper structure in the conversation. The example 3.2 portrays how the first rubrics expresses the mismatch between user requests and AI responses but fails to explain why this is problematic or how it impacts the interaction. In contrast, the second statement, framed with Grice's maxim of *relevance*, provides a more thorough and fundamental explanation of the communication breakdown.

Example 3.2. The following rubrics below express the same idea:
Without CDP: *The assistant does not adhere to the user's specific request for the format or structure of the response.*
With CDP: *The assistant's responses are not aligned with the user's expectations.*

Domain Sensitization & Rubric Complexity (DS). The prompt incorporates instructions for the LLM to focus on the aspects that make a conversation positive/negative in the cognizance of the domain. For the Debugging use case, this includes progress towards the debugging task, localization of the bug, answering user queries, or providing an outright code snippet as the solution. Given that very complex rubrics are difficult to reason about and often lead to non-determinism in response even at low-temperature levels, we restrict the assertions to be short and clean. The instruction prompt limits the generation of complex rubrics by a natural language insight that simple assertive statements tend to have a single verb.

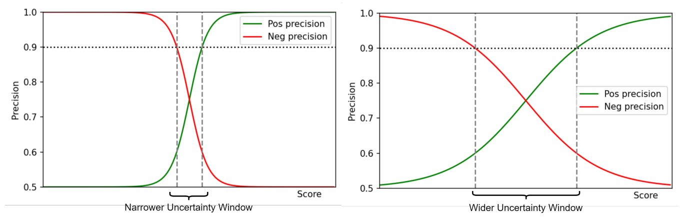


Figure 4: Positive & Negative Precision vs threshold for two hypothetical rubric sets. A narrower uncertainty window provides for larger ranges for high-confidence labels.

3.3 Selecting Optimized Rubrics

The generation procedure from Section 3.2 generates a large pool of rubrics $\{r_1, \ldots, r_n\}$. Here, we select a subset of rubrics $\mathsf{R_S}$ from this pool that is both of a reasonable size and that is usable in practice. We want $\mathsf{R_S}$ to correlate with the training data, i.e., it should score positive conversations higher than negative conversations. Going from a real-valued score given by a set of rubrics to a boolean positive-negative classification requires selecting a threshold θ–scores over and under θ will be classified as positive and negative, respectively. However, selecting a threshold is often problematic: varying the threshold slightly can lead to different results, and the selection is often ad-hoc. To aid with this issue, we propose a selection technique that automatically produces a robust pair of rubric-set and threshold.

Correctness and correctness loss. Given a set of rubric R and a conversation c, we first define the score $\mathsf{Score}(\mathsf{R}, c)$ to be the sum $\sum_{r \in \mathsf{R}} \mathsf{Eval}_r(c)$ where $\mathsf{Eval}_r(c)$ is the normalized $[0, 10]$ Likert score for a single rubric and conversation described in Section 3.2. Intuitively, R should separate positive and negative conversations, i.e., positive conversations should be scored significantly higher than negative conversations. Hence, we define the *correctness loss* $\mathsf{Loss_C}(\mathsf{R})$ to be: $\mathsf{Loss_C}(\mathsf{R}) = \mathsf{mean}_{c \in \mathsf{C}_+} \mathsf{Score}(\mathsf{R}, c) - \mathsf{mean}_{c \in \mathsf{C}_-} \mathsf{Score}(\mathsf{R}, c)$

Sharpness and Sharpness loss. Apart from correctness, we want the rubric-set R to be easily convertible into a binary classifier using a threshold. The correctness loss from above only accounts for

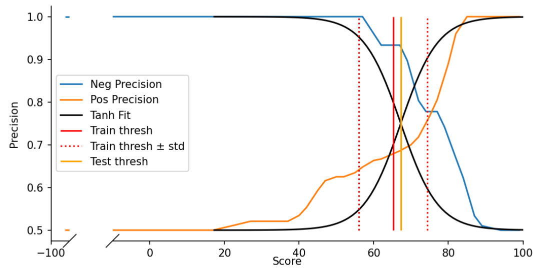


Figure 5: Positive & Negative Precision v Score(Threshold) plot for RUBICON, comparing the $Tanh$ curves centered at test threshold and the corresponding precision curves post selection.

having a high average separation of score between positive and negative conversations but does not account for the *sharpness of the separation.* To understand the notion of sharpness, let us first define some basic terminology. Given a threshold θ, we call a conversation c *positive* if $\text{Score}(R, c) > \theta$, and *negative* otherwise. The *positive precision* $\text{Prec}_+(R, \theta)$ (resp. *negative precision* $\text{Prec}_-(R, \theta)$) is defined as the fraction of conversations that are labelled as positive (resp. negative) that are also positive (resp. negative) by ground truth. There is a trade-off between Prec_+ and Prec_-: we can get very high positive precision by setting a high threshold (i.e., labelling almost all conversations negative) and vice-versa.

Figure 4 shows two hypothetical plots of threshold against positive and negative precision for two hypothetical rubric sets R_1 and R_2. The plot on the left shows a single narrow range of candidate thresholds which gives high positive and negative precision, while the one on the right does not.

The *sharpness loss* quantifies this idea of sharpness of a rubric-set precisely: we fit a tanh function for both positive and negative precision curves while ensuring the mid-point θ_c for both curves is the same. Given a rubric-set R, let

$$\theta_c = \frac{1}{2} \cdot \left(\frac{\sum_{c \in C_+} \text{Score}(R, c)}{|C_+|} + \frac{\sum_{c \in C_-} \text{Score}(R, c)}{|C_-|} \right)$$

We define the sharpness loss as:

$$\begin{aligned} \text{Loss}_S(R) = {} & \text{SOSError}_\theta[\tanh(\theta - \theta_c), \text{Prec}_+(R, \theta)] \\ & + \text{SOSError}_\theta[\tanh(-(\theta - \theta_c)), \text{Prec}_-(R, \theta)] \end{aligned}$$

Note that we elide from the notation the scaling factors that ensure that both the precision curves and tanh are in the same range.

Iterative selection of rubrics. Algorithm 1 depicts an algorithm for selecting a set of rubrics R_S from a rubric pool R_P. It proceeds by starting with an empty set R_S (line 1), and iteratively growing the set by adding the rubric r^* that minimizes the (weighted) sum of correctness and sharpness loss as defined above (lines 2- 4). The weight α (line 11) is a hyperparameter selected based on the best average performing configuration in a 5-fold cross-validation over the training conversation data.

In practice, we also modify Algorithm 1. Rather than a combined budget n, we instead have separate n_{SAT} and n_{DSAT} budgets for the SAT & DSAT rubrics as defined in Section 3.2. We stop selecting either SAT or DSAT rubrics as soon as the respective budget is hit.

Algorithm 1 Selecting rubrics from rubric pool

```
Require: Training dataset of conversations $C = C_+ \cup C_-$ partitioned into
    positive and negative conversations
Require: Rubric pool $R_P$
Require: Rubric budget $n$
Require: Sharpness loss weight $\alpha \in \mathbb{R}^{>0}$
Ensure: Selected rubric set $R_S$
Ensure: Threshold $\theta$
 1: $R_S \leftarrow \emptyset$
 2: while $|R_S| \neq n \wedge |R_P \setminus R_S| > 0$ do
 3:     $r^* \leftarrow \arg\max_{r \in R_P \setminus R_S} \text{Loss}(R_S \cup \{r\}, C_+, C_-)$
 4:     $R_S \leftarrow R_S \cup \{r^*\}$
 5: return $\langle R_S, \text{GetThreshold}(R_S, C_+, C_-) \rangle$
 6:
 7: procedure Loss(R, $C_+$, $C_-$)
 8:     $LC \leftarrow \text{mean}_{c \in C_+} \text{Score}(R, c) - \text{mean}_{c \in C_-} \text{Score}(R, c)$
 9:     $\theta_c \leftarrow \text{GetThreshold}(R, C_+, C_-)$
10:     $LS \leftarrow \text{Loss}_S(R, \theta_c, C_+, C_-)$
11:     return $LC + \alpha \cdot LS$
12:
13: procedure GetThreshold(R, $C_+$, $C_-$)
14:     return $\frac{1}{2} \cdot (\text{mean}_{c \in C_+} \text{Score}(R, c) + \text{mean}_{c \in C_-} \text{Score}(R, c))$
```

4 EVALUATION

In order to evaluate RUBICON, we aim to answer the following research questions:

- **RQ1:** How does the effectiveness of RUBICON compare to the other rubric-backed baseline methods?
- **RQ2:** What is the extent of the impact of Domain Sensitization (DS) and Conversation Design Principles (CDP) instructions on the performance of RUBICON?
- **RQ3:** How does the effectiveness of the proposed selection policy compare to other baselines in selecting the final set of rubrics?
- **RQ4:** How does the effectiveness of Numeric values for rubrics compare to that of Likert Scales?

4.1 Data

Collection & Sanitization. The conversation data was sourced from a C# Debugger Copilot assistant deployed in an IDE at a large software company that was mined over 30 calendar days of deployment. All conversations have a unique Conversation ID and were collected anonymously with no link back to the user. Of 127 conversations, after clean-up of logging & data corruption issues, we have a total of 100 conversations carried out by actual users of the debugging copilot. The deployed assistant is aware of the context around the exceptions like the error message, stack trace, etc., and source code details like exception location, current file etc. The deployment was controlled and limited to maintain user access across the spectrum of proficiency and experience in software development in C#.

Annotation. The filtered set of 100 conversations are then annotated for ground truth of being *Positive* or *Negative* in reference to concrete aid to the user with the debugging task. Two authors with experience of over 2 & 4 years of professional software development in the industry annotate the data in the binary classification task. In the initial phase, the first 20 conversations were annotated together while discussing themes for annotation. Post these insights, both engineers annotated another set of 20 conversations independently, which were then tested for Inter-Rater Reliability (IRR). We found that the annotators agreed on 17 conversations, giving us an IRR of 0.85, which is an 'almost perfect' agreement. We split the remaining set between both the raters to label independently while communicating in case of doubts or need for discussion.

Split. Given the limited data availability, we carry out a 50:50 train-test split on the filtered conversation data. Both classification labels are homogeneously represented in the splits.

4.2 Metrics

Accuracy, Precision, Recall and F1 Score are calculated on the test set C_{test} based on threshold θ learned from the C_{train} for each final set of rubrics. We define two additional metrics of $\Delta NetSAT$ and $YieldRate$.

$\Delta NetSAT$ score measures how well a rubric set separates the clusters of positive and negative conversations across the score axis. The metric is the difference between the mean of NetSAT score of positive conversations and mean of NetSAT score of the negative conversations i.e. $\Delta NetSAT = \overline{NetSAT}_{+} - \overline{NetSAT}_{-}$

The higher the $\Delta NetSAT$, the higher the separation between the two distributions, which means that the rubric is able to differentiate positive and negative conversations better.

Yield Rate measures what fraction of the test set C_{test} can be labelled with a certain precision. Classifying conversations as distinctly positive or negative is a difficult and subjective task, even for humans. To be confident about our generated labels, we only consider conversations which can be classified with a minimum acceptable precision. $YieldRate@P$ is the percentage of conversations that fall above the P precision threshold. For the purpose of our experiments, we fix P to be 0.9 and refer to the metric as $YieldRate@90$. We calculate the Prec_{-} and Prec_{+} as a function of threshold θ. $\text{Prec}_{-}(\theta)$ is the percentage of negative labels classified with a threshold θ, similarly for Prec_{+}. By setting a 0.9 precision threshold, we divide the scores into three windows. The first window with $\text{Prec}_{-} > 0.9$, the second with $\text{Prec}_{-} < 0.9$ and $\text{Prec}_{+} < 0.9$, which we call the uncertainty window, and the third with $\text{Prec}_{+} > 0.9$. Only the conversations with scores outside of the uncertainty window are considered, and the yield rate measures the ratio of these conversations. Figure 4 shows the positive and negative precision curves based on a 0.9 precision cutoff. *Note*: A high overall precision does not translate to high Yield Rate as the latter depends on the spread of the score and the uncertainty window. One may have high precision values for more data with low confidence in the uncertainty interval; however, this performance does not translate to practical deployment scenarios.

4.3 Baselines

We compare the RUBICON against the other baselines for the end-to-end task of generating score-able rubric sets. We further compare dedicated baselines for Rubric Selection policy separately.

BING. SPUR investigates and generates rubrics for the Bing Copilot [16] generated with the SPUR in their paper. These SAT/DSAT rubrics were learned from training data for open-domain question-answering of Bing Copilot.

BING (Domain Adapted). For this baseline, we manually adapted the *BING* rubrics to be more inclusive of our domain and expectations. This is accomplished by an independent Software Engineer who works on the Debugger Copilot with over 6 years of professional development experience. We ask them to *completely* replace some rubrics that did not apply to the use case & the intent of Copilot with some *domain-aware* rubrics for both SAT/DSAT based on their intuition and understanding. The $\boldsymbol{BING_{DA}}$ rubric set finally received had 3 replacements in SAT and 4 replacements in DSAT.

SPUR. We re-implemented the SPUR technique [16] using the prompts and strategies provided in the paper. We ran this pipeline on our training data to get the final set of rubrics for evaluation.

4.3.1 Selection Policy. Given a rubric pool and some validation data, the selection policy chooses a subset of rubrics optimised for performance on the validation dataset.

UCB Bandits. This policy treats rubric selection as a bandit selection problem, and UCB has been explored as a solution to select a subset of prompts based on their performance [25]. We define the reward score of a rubric as the sum of scores it receives over the respective batch of conversations. Reinforcement Learning methods assume that prompt evaluations are limited and costly, creating a trade-off between exploration and exploitation. However, this may not necessarily be the case in domain-specific conversations where data may be scarce, rendering evaluating all rubrics over the smaller data size manageable.

Brute Force. We also compare this with a brute force approach where the score of each rubric is defined the same as in UCB Bandits. This assumes that complete evaluation over the rubric and conversation is manageable, akin to RUBICON.

4.4 Experimental Setup

We use GPT-4 to power RUBICON and SPUR. Two authors of the paper created the domain sensitization instructions as discussed in 3 (exact instructions in [2]) For all these configurations, we maintain a consistent selection policy for the final rubrics, as proposed in algorithm 1, with both SAT & DSAT rubric set to $N = 10$.

To evaluate RQ2, we conduct an ablation study to differentiate the contribution of each component in the prompts. We establish four configurations by systematically excluding instructions related to DS, CDP, and both from the final proposed prompts.

For RQ3, For all selection policies, we fix the rubric pool to the one acquired on executing the augmentation as described in 3.2.

Finally, for RQ4, we take rubrics generated from SPUR baseline and score them with different evaluators.

4.5 Results

4.5.1 RQ1: How does the effectiveness of RUBICON *compare to the other rubric-backed baseline methods?* Table 1 summarises the results. RUBICON demonstrates superior performance across all metrics except for precision, where $BING_{DA}$ was slightly superior. The improvement of RUBICON over other methods is readily apparent in $\Delta NetSAT$. This highlights the effectiveness of our tool in distancing positive conversations from negative ones in terms of $NetSAT$ which creates the highest separation as compared to other baselines. In particular, SPUR, which is unable to generalize over our data to create differences in scores of positive and negative conversations. Furthermore, we observed that RUBICON also outperforms in terms of $YR@90$, scoring 20 points (abs.) higher than the next best. This indicates that our tool offers a high precision (>0.9) in classifying 84% of conversations. SPUR only classifies 28% of conversations with this level of precision. While $BING_{DA}$ achieved

Technique	ΔNS	$YR@90$	A	P	R	$F1$
$BING$	11.0	58.0	70.0	64.7	88.0	74.6
$BING_{DA}$	11.3	64.0	76.0	**70.9**	88.0	78.6
$SPUR$	3.0	28.0	58.0	55.9	76.0	64.4
$\boldsymbol{RUBICON}$	**27.4**	**84.0**	**76.0**	68.6	**96.0**	**80.0**

Table 1: Rubric performance across different techniques. $\Delta NS = \Delta NetSAT$, $YR@90 = YieldRate@90\%, A = Accuracy, P = Precision, R = Recall, F1 = F1score$

a slightly higher precision score, it classified far fewer conversations (64%) with the minimum acceptable level of precision. Our results suggest that RUBICON rubrics perform better than SPUR rubrics and rubrics manually written by experts.

4.5.2 RQ2: What is the extent of the impact of Domain Sensitization (DS) and Conversation Design Principles (CDP) instructions on the performance of RUBICON?.

Table 2 showcases the various configurations for the ablation of the augmentation step. We observe that domain sensitization plays a crucial role in recognizing patterns and thus significantly improves metrics across the board. Notably, the removal of both DS and CDP resulted in the steepest decline, confirming that these components play a pivotal role in the proposed technique. This also suggests that DS and CDP are complementary in nature, and their combination optimizes the performance of our technique. A higher recall in the absence of DS could indicate over-generalization, leading to more true positives but also more false positives. This means that DS helps the RUBICON recognize more nuanced and specific patterns within the domain, thus becoming more selective in identifying positive instances.

Technique	ΔNS	$YR@90$	A	P	R	$F1$
$\boldsymbol{RUBICON}$	**27.4**	**84.0**	**76.0**	**68.6**	96.0	**80.0**
$-CDP$	23.7	72.0	64.0	58.5	96.0	72.7
$-DS$	17.9	66.0	52.0	51.0	**100.0**	67.6
$-CDP - DS$	15.7	62.0	50.0	50.0	**100.0**	66.67

Table 2: Ablation study of Rubric Augmentation

4.5.3 RQ3: How does the effectiveness of the proposed selection policy compare to that of other baselines in selecting the final set of rubrics?

The experimental results, as shown in Table 3, reveal several key insights about the performance of our selection policy. Firstly, our policy outperforms all baseline methods in terms of $\Delta NetSAT$ score and $YieldRate@90\%$ precision with clear contributions from both components of loss as hypothesized. These results suggest that our selection policy is particularly effective in separating positive and negative conversations while also being able to confidently label a larger proportion of conversations compared to the baselines. Interestingly, even though $BruteForce$ and $CorrectnessLoss$ consider all possibilities, the objective allows the latter to have a better accuracy and separation while maintaining a similar confidence. The last row suggests that $SharpnessLoss$ in RUBICON enables us to classify confidently without losing on performance as compared to other baselines with respect to the $NetSAT$ separation, counter balancing effects of overfitting tendencies from the other loss component.

Technique	ΔNS	$YR@90$	A	P	R	$F1$
UCB	23.6	68.0	72.0	65.7	92.0	76.7
BruteForce	24.2	74.0	74.0	67.7	92.0	78.0
CorrectnessLoss	**29.7**	74.0	**76.0**	**69.7**	92.0	79.3
RUBICON	27.4	**84.0**	**76.0**	68.6	**96.0**	**80.0**

Table 3: Comparative analysis of Selection policies.

4.5.4 RQ4: How does the effectiveness of Numeric values for rubrics compare to that of Likert Scales?

Our experiment results, as shown in Table 4, demonstrated that the Likert based evaluator significantly improved the performance of rubrics from SPUR across all performance metrics. This reinforces the ability of LLMs to perceive Agreement/Disagreement better than numerical values for the purpose of rubric or assertion scoring. However, the experiment also revealed an important insight: while the Likert scoring system can enhance performance, incorporating domain-specific knowledge into the rubric evaluation process can yield even better performance improvements for the downstream task.

Evaluator	ΔNS	$YR@90$	A	P	R	$F1$
$Numeric$	3.0	28.0	58.0	55.9	76.0	64.4
Likert	14.5	48.0	70.0	67.9	76.0	71.7
Likert$_{DA}$	**16.1**	**58.0**	**80.0**	**77.8**	**84.0**	**80.8**

Table 4: Likert scale vs Numeric scale

5 THREATS TO VALIDITY

Internal Validity The ground truth labels for the binary classification of conversations into positive or negative were manually assigned. Despite our high inter-annotator agreement, this process is inherently subjective and prone to errors or inconsistencies, which might impact the rubric learning process.

External Validity The data used in our experiments were collected over a month from a C# debugger copilot assistant deployed in an IDE at a large software company. The limitation in the dataset's diversity in terms of the number of conversations, we could collect due to ethical limitations might lead to biases in the results. Additionally, our study is conducted in the context of software debugging tasks, making the results to domain-specific to software engineering (SE). We make debugging specific sensitization in our experiments and the resultant rubrics and evaluator may not directly translate to other domains or tasks. While the concept and technique are transferable, generalizing the results to other areas within SE or other domains might require additional research and validation.

Construct Validity Our research leverages automated scoring of rubrics based on instructions tuned LLM. However, the performance of the automated scoring is optimized with respect to the task, its accuracy is not directly evaluated. Changes to the underlying scoring model including inherent training biases could influence our results.

The use of Likert scale and its arithmetic manipulations make several implied assumptions. We assume the difference between 'Neutral' and 'Disagree' to be numerically the same as 'Agree' and 'Strongly Agree', since Likert is an ordinal scale, and we convert it to [0-10] (arbitrary) scale. We also do not have a separate category for 'Not Applied', and it is clubbed with 'Neutral'.

6 CONCLUSION

We introduced RUBICON, a novel methodology for automated evaluation of AI-assisted debugging conversations. We've shown that by incorporating domain knowledge and conversational design principles, we can considerably enhance the quality of generated rubrics. Furthermore, we've demonstrated that our proposed selection policy effectively differentiates between good and bad conversations and surpasses baseline methods. A domain-adapted Likert scale scoring system also proved effective in scoring rubrics. We share the prompts used in the pipeline as well as the rubrics generated at [2]

RUBICON has been successfully deployed in a popular IDE at a large software company to monitor two Software Engineering focused AI Assistant Copilots, with promising results. The insights gained from this study open new avenues for future research, emphasizing that the synergistic application of AI and human expertise can lead to robust and effective tools for evaluating and improving conversational debugging experiences.

REFERENCES

[1] Open AI. [n. d.]. ChatGPT. https://chat.openai.com

[2] Anonymous Authors. [n. d.]. Supplementary Material for RUBICON. https://docs.google.com/document/d/e/2PACX-1vTqlC7TrNnJx_yfJtNuul-FGYa30vCeWIxK6N7M0EhIu1aZLTtXxCTOtGfDH8Hmw_Gqwjg_ieBLW5rA/pub

[3] Praveen Kumar Bodigutla, Lazaros Polymenakos, and Spyros Matsoukas. 2019. Multi-domain Conversation Quality Evaluation via User Satisfaction Estimation. In *NeurIPS 2019 Workshop on Conversational AI*. https://www.amazon.science/publications/multi-domain-conversation-quality-evaluation-via-user-satisfaction-estimation

[4] Bhavya Chopra, Yasharth Bajpai, Param Biyani, Gustavo Soares, Arjun Radhakrishna, Chris Parnin, and Sumit Gulwani. 2024. Exploring Interaction Patterns for Debugging: Enhancing Conversational Capabilities of AI-assistants. *arXiv preprint arXiv:2402.06229* (2024).

[5] Ondrej Dusek, Jekaterina Novikova, and Verena Rieser. 2017. Referenceless Quality Estimation for Natural Language Generation. *ArXiv* abs/1708.01759 (2017). https://api.semanticscholar.org/CorpusID:37763454

[6] Wieland Eckert, Esther Levin, and R. Pieraccini. 1998. User modeling for spoken dialogue system evaluation. 80 – 87. https://doi.org/10.1109/ASRU.1997.658991

[7] Ryan Fellows, H. Ihshaish, Steve Battle, Ciaran Haines, Peter Mayhew, and J. Ignacio Deza. 2021. Task-oriented Dialogue Systems: performance vs. quality-optima, a review. *ArXiv* abs/2112.11176 (2021). https://api.semanticscholar.org/CorpusID:245353758

[8] Nah Fiona Fui-Hoon, Zheng Ruilin, Cai Jingyuan, Siau Keng, and Chen Langtao. 2023. Generative AI and ChatGPT: Applications, challenges, and AI-human collaboration. *Journal of Information Technology Case and Application Research* 25, 3 (2023), 277–304. https://doi.org/10.1080/15228053.2023.2233814

[9] Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: core tasks, applications and evaluation. *J. Artif. Int. Res.* 61, 1 (jan 2018), 65–170.

[10] Marco Gerosa, Bianca Trinkenreich, Igor Steinmacher, and Anita Sarma. 2024. Can AI serve as a substitute for human subjects in software engineering research? *Automated Software Engg.* 31, 1 (jan 2024), 12 pages. https://doi.org/10.1007/s10515-023-00409-6

[11] GitHub. 2023. GitHub Copilot. https://github.com/features/copilot

[12] Ilya Kulikov, Alexander H. Miller, Kyunghyun Cho, and Jason Weston. 2018. Importance of a Search Strategy in Neural Dialogue Modelling. *ArXiv* abs/1811.00907 (2018). https://api.semanticscholar.org/CorpusID:53297919

[13] Sivaraju Kuraku, Fnu Samaah, Dinesh Kalla, and Nathan Smith. 2023. Study and Analysis of Chat GPT and its Impact on Different Fields of Study. (03 2023).

[14] Margaret Li, Jason Weston, and Stephen Roller. 2019. ACUTE-EVAL: Improved Dialogue Evaluation with Optimized Questions and Multi-turn Comparisons. *ArXiv* abs/1909.03087 (2019). https://api.semanticscholar.org/CorpusID:202538657

[15] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013

[16] Ying-Chun Lin, Jennifer Neville, Jack W. Stokes, Longqi Yang, Tara Safavi, Mengting Wan, Scott Counts, Siddharth Suri, Reid Andersen, Xiaofeng Xu, Deepak Gupta, Sujay Kumar Jauhar, Xia Song, Georg Buscher, Saurabh Tiwary, Brent Hecht, and Jaime Teevan. 2024. Interpretable User Satisfaction Estimation for Conversational Systems with Large Language Models. arXiv:2403.12388 [cs.IR]

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[18] Clara Meister and Ryan Cotterell. 2021. Language Model Evaluation Beyond Perplexity. arXiv:2106.00085 [cs.CL]

[19] Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2018. RankME: Reliable Human Ratings for Natural Language Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 72–78. https://doi.org/10.18653/v1/N18-2012

[20] OpenAI. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[21] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Pierre Isabelle, Eugene Charniak, and Dekang Lin (Eds.). Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[22] Chris Parnin, Gustavo Soares, Rahul Pandita, Sumit Gulwani, Jessica Rich, and Austin Z. Henley. 2023. Building Your Own Product Copilot: Challenges, Opportunities, and Needs. arXiv:2312.14231 [cs.SE]

[23] Cathy Pearl. 2016. *Designing Voice User Interfaces: Principles of Conversational Experiences* (1st ed.). O'Reilly Media, Inc.

[24] Diana Perez-Marin and Ismael Pascual-Nieto. 2011. *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA.

[25] Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 7957–7968. https://doi.org/10.18653/v1/2023.emnlp-main.494

[26] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

[27] Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? How controllable attributes affect human judgments. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 1702–1723. https://doi.org/10.18653/v1/N19-1170

[28] Olga Seminck. 2023. Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots by Michael McTear. *Computational Linguistics* 49, 1 (March 2023), 257–259. https://doi.org/10.1162/coli_r_00470

[29] Sakib Shahriar and Kadhim Hayawi. 2023. Let's Have a Chat! A Conversation with ChatGPT: Technology, Applications, and Limitations. *Artificial Intelligence and Applications* 2, 1 (June 2023), 11–20. https://doi.org/10.47852/bonviewaia3202939

[30] Itamar Shatz. [n. d.]. Grice's Maxims of Conversation: The Principles of Effective Communication. https://effectiviology.com/principles-of-effective-communication/

[31] Usneek Singh, Piyush Arora, Shamika Ganesan, Mohit Kumar, Siddhant Kulkarni, and Salil Rajeev Joshi. 2024. Comparative Analysis of Transformers for Modeling Tabular Data: A Casestudy using Industry Scale Dataset. In *Proceedings of the 7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD)* (Bangalore, India) *(CODS-COMAD '24)*. Association for Computing Machinery, New York, NY, USA, 449–453. https://doi.org/10.1145/3632410.3632456

[32] Eric Smith, Orion Hsu, Rebecca Qian, Stephen Roller, Y-Lan Boureau, and Jason Weston. 2022. Human Evaluation of Conversations is an Open Problem: comparing the sensitivity of various methods for evaluating dialogue agents. In *Proceedings of the 4th Workshop on NLP for Conversational AI*, Bing Liu, Alexandros Papangelis, Stefan Ultes, Abhinav Rastogi, Yun-Nung Chen, Georgios Spithourakis, Elnaz Nouri, and Weiyan Shi (Eds.). Association for Computational Linguistics, Dublin, Ireland, 77–97. https://doi.org/10.18653/v1/2022.nlp4convai-1.8

[33] Abhinav Srivastava, Amlan Kundu, Shamik Sural, and Arun Majumdar. 2008. Credit Card Fraud Detection Using Hidden Markov Model. *IEEE Transactions on Dependable and Secure Computing* 5, 1 (2008), 37–48. https://doi.org/10.1109/TDSC.2007.70228

[34] Elior Sulem, Omri Abend, and Ari Rappoport. 2018. BLEU is Not Suitable for the Evaluation of Text Simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Ellen Riloff, David Chiang,

Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, Brussels, Belgium, 738–744. https://doi.org/10.18653/v1/D18-1081

[35] Stefan Ultes, Paweł Budzianowski, Iñigo Casanueva, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei hao Su, Tsung-Hsien Wen, Milica Gašić, and Steve J. Young. 2017. Domain-Independent User Satisfaction Reward Estimation for Dialogue Policy Learning. In *Interspeech*. https://api.semanticscholar.org/CorpusID:649649

[36] Yequan Wang, Jiawen Deng, Aixin Sun, and Xuying Meng. 2023. Perplexity from PLM Is Unreliable for Evaluating Text Quality. arXiv:2210.05892 [cs.CL]

[37] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too?. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Iryna Gurevych and Yusuke Miyao (Eds.). Association for Computational Linguistics, Melbourne, Australia, 2204–2213. https://doi.org/10.18653/v1/P18-1205

A APPENDIX

We provide the prompts we use for generating our rubrics used and evaluated in our experiments.

A.1 Supervised Extraction Prompt

```
Your job is to understand and elaborate on the signals
     of a conversation which are indicative of a **good
     ** conversation. The conversation is between a user
      and an AI debugging assistant designed to work on
     codebases with the task of debugging an exception
     for the user. The AI is designed to hold a
     conversation to understand, ask for more
     information, and investigate the bug, then provide
     solutions to aid the user in their debugging tasks.
      You will be given a conversation that a user had
     with an AI agent where the user provided a signal
     of satisfaction through a like button.

You should think of what constitutes a good
     conversation, where the user makes progress in
     their task, and summarize how a user expresses that
      they are **satisfied** with their interaction with
      an AI agent. You should consider how much either
     party takes active steps to achieve their goals in
     the conversation, and compare it with an idealized
     version of the conversation. Your summary on the
     good characteristics and user satisfaction of the
     conversation will be later used to generate a
     diverse and holistic feedback on the conversation,
     so focus on the factors that determine the
     interaction's success or failure.

Your task is to summarize signals indicative of a good
     conversation.
Instructions:
- First write a paragraph summarizing the conversation
     and highlighting the moment where the user would
     feel satisfied with the interaction.
- Following that, provide your answer in xml format
     between <REASONS></REASONS> tags.
- Return NONE if you can't think of any part of the
     conversation that indicates a good conversation.
- The reasons you summarized should be grounded on the
     conversation history only. You should **NOT**
     extrapolate, imagine, or hallucinate beyond the
     text of the conversation that is given.
- The reasons should be mutually exclusive and simple.
- Your summary should be concise, use bullet points,
     and provide no more than 3 reasons.
- Your reasons can begin with "The user ..." or "The
     assistant ...". The user's responses can also
     contribute to a good conversation, and if present,
     you must capture that aspect as well.

<CONVERSATION>
{conversation}
</CONVERSATION>
The main reasons why the interaction is a good
     conversation are:
```

A.2 Rubric Summarization Prompt

```
# Task
Your job is to summarize why an interaction between a
     user and an AI is a **good** conversation and
     provide a rubric for evaluation of a single
     conversation. You will be given a list of example
     explanations from conversations that users had with
      an AI agent.

# Instruction
Your task is to provide a rubric to identify a user's
     expectations and requirements, and how much the AI
     was able to understand and meet them. You must
     think how much either parties take active steps to
     achieve their goals in the conversation.

Generate the rubrics based on the following maxims of
     what an ideal conversation is supposed to look like
     :
    1. The maxim of quantity, where one tries to be as
       informative as one possibly can, and gives as
       much information as is needed, and no more.
    2. The maxim of quality, where one tries to be
       truthful, and does not give information that is
        false or that is not supported by evidence.
    3. The maxim of relation, where one tries to be
       relevant, and says things that are pertinent to
        the discussion.
    4. The maxim of manner, when one tries to be as
        clear, as brief, and as orderly as one can in
        what one says, and where one avoids obscurity
        and ambiguity.
As the maxims stand, there may be an overlap, as
    regards the length of what one says, between the
    maxims of quantity and manner; this overlap can be
    explained (partially if not entirely) by thinking
    of the maxim of quantity (artificial though this
    approach may be) in terms of units of information.
    In other words, if the listener needs, let us say,
    five units of information from the speaker, but
    gets less, or more than the expected number, then
    the speaker is breaking the maxim of quantity.
    However, if the speaker gives the five required
    units of information, but is either too curt or
    long-winded in conveying them to the listener, then
     the maxim of manner is broken.

# Example Explanations of Good Conversations
{self_reflection_cases}

# Previous Rubrics
{documentation_description}

# Now summarize these examples into a rubric to
    identify a good conversation. Requirements:
* Provide your answer as a numbered list of up to {
    num_rubric} bullet items.
* The rubric should be user-centric, concise, and
    mutually exclusive.
* The rubric should be mutually exclusive with respect
    to the previous rubrics as well. If a point is
    already covered in the previous rubrics, you must
    skip it in the generated rubric.
* The rubric should **NOT** directly refer to the
    maxims.
* The rubric should consist of simple sentences. Each
    item must contain only one verb.
* Keep making the rubric diverse enough so that it
    covers most of the characteristics of good
    conversations.
* Provide your answer as a numbered list of bullet
    items in <Rubric></Rubric>. The output format is as
     follows:
```
# Output
<Rubric>
1. [item 1]
2. [item 2]
3. [item 3]
...
</Rubric>
```

# Output
```

A.3 Rubric Evaluation Prompt

```
You are a highly skilled technical conversation
    evaluation system designed to evaluate
    conversations between users and an AI copilot
    assistant for debugging tasks.
You are tasked to go over a text or C# code from Visual
     Studio conversation and then give a specific
    answer to questions about the reference
    conversation.

There are only two parties in the conversation:
USER: Is a human software developer who has encountered
     a bug in their code.
ASSISTANT: Is an AI code debugging agent trained
    specifically to diagnose, investigate and assist
    towards fixing an issue.

A roughly ideal conversation should look like this:
USER gives description of a bug.
ASSISTANT tries to diagnose the error and provide steps
     to the USER to solve/ investigate the issue.
The USER can ask follow up questions or reply back to
    any suggestions or information requests from the
    ASSISTANT.
The overall goal of the conversation is to reach a
    point where you can isolate and create a fix for
    the given bug.

Here are a few things you should keep in mind:
    1. You must think step by step, and first justify
        how you approach the answer before selecting
        your final option.
    2. Then, you select a final answer, which is
        selected based on the justification you
        presented. The final answer is always from the
        list of possible answers provided along with
        each question in English, irrespective of the
        conversation being in any other language.
    3. The conversations are tagged with "USER" for the
         human developer and "ASSISTANT" for the AI
        assistant.
    4. Your answer evaluates the conversation
        objectively, without any assumptions.
    5. The shared conversation starts right from the
        first prompt given by the user.
    6. You are not the AI assistant, you are a third
         party independent evaluator.
    7. If the question is not applicable answer with '
         Neutral' option.

MANDATORY RESPONSE FORMAT:
```
```
1. <Question 1 without the options>
<only one option from the list> Strongly Disagree/
     Disagree/Neutral/Agree/Strongly Agree
2. <Question 2 without the options>
<only one option from the list> Strongly Disagree/
     Disagree/Neutral/Agree/Strongly Agree
...
<end>
```
```
Each question should only take up two lines.

Given the conversation, answer the following questions
     with just one option from the given list which MUST
      be followed by the end marker "<end>":
```

A.4 Semantic Deduplication Prompt (Post processing)

```
You are an AI assistant that processes assertion
     rubrics. Each assertion is a question with respect
     to a conversation. A rubric must be concise, well-
     rounded, and all assertions must be mutually
     exclusive in their ideas.

Return the below set of rubrics after removing the
     duplicate assertions. Remove anything which are
     similar in ideas:
<RUBRICS>
```