Exploring User Dissatisfaction: Taxonomy of Implicit Negative Feedback in Virtual Assistants

Moushumi Mahato[1], **Avinash Kumar**[2], **Kartikey Singh**[3], **Javaid Nabi**[4], **Debojyoti Saha**[5], **Krishna Singh**[6]
Language AI Appliances & Analytics
Samsung R&D Institute, Bengaluru, India
{[1]moushumi.m, [2]avinash1.k, [3]kartikey.s, [4]javaid.nabi, [5]d.saha, [6]krish.singh}@samsung.com

Abstract

The success of virtual assistants relies on continuous performance monitoring to ensure their competitive edge in the market. This entails assessing their ability to understand user intents and execute tasks effectively. While user feedback is pivotal for measuring satisfaction levels, relying solely on explicit feedback proves impractical. Thus, extracting implicit user feedback from conversations of user and virtual assistant is a more efficient approach. Additionally, along with learning whether a task is performed correctly or not, it is extremely important to understand the reasons behind any incorrect execution. In this paper, we introduce a framework designed to identify dissatisfactory conversations, systematically analyze these conversations, and generate comprehensive reports detailing the reasons for user dissatisfaction. By implementing a feedback classifier, we identify conversations that indicate user dissatisfaction, which serves as a sign of implicit negative feedback. To analyze negative feedback conversations more deeply, we develop a lightweight pipeline called an issue categorizer ensemble with multiple models to understand the reasons behind such dissatisfactory conversations. We subsequently augment the identified discontented instances to generate additional data and train our models to prevent such failures in the future. Our implementation of this simple framework, called AsTrix (Assisted Triage and Fix), led to significant enhancements in the performance of our smartphone-based In-House virtual assistant, with successful task completion rates increasing from 83.1% to 92.6% between June 2022 and March 2024. Moreover, by automating the deeper analysis process targeting just five major issue types contributing to the dissatisfaction, we significantly address approximately 62% of the negative feedback conversation data.

1 Introduction

The efficacy of virtual assistants (VAs) depends upon their ability to accurately interpret user inputs and execute appropriate tasks. While assessing their success rate is crucial, it is equally important to scrutinize their failure rate, as this presents opportunities for improvement. Hence, continuous evaluation of the virtual assistant's performance plays an important role in the system's improvements. There has been abundant amount of work done to evaluate virtual assistants (Liang et al., 2021), (Kachuee et al., 2020), (Pan et al., 2022), (Park et al., 2020), (Shen et al., 2022), (Cai and Chen, 2020), (Hu et al., 2023), (Deng et al., 2022), (Song et al., 2023), (Ye et al., 2023) but most of them is related to taking reference from user feedback data (click data, data with likes/dislikes) to derive implicit feedback information (Pan et al., 2022), (Park et al., 2020), (Shen et al., 2022), (Lin et al., 2024). The availability of explicit feedback data may be challenging and may be restricted to only a few data patterns. Moreover, most of this work focused on estimating user satisfaction and improving classification accuracy, overlooking the interpretability.

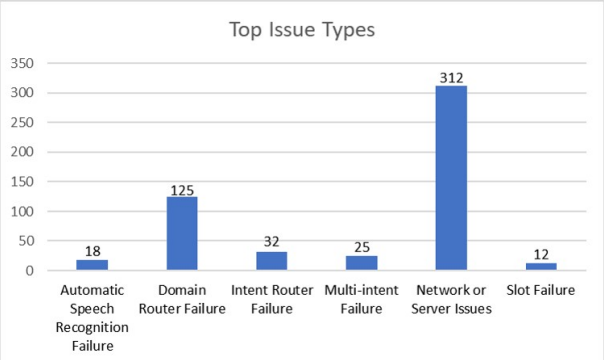

Figure 1: Top issues identified in In-House-VA between 2022-10-07 and 2022-10-19

In this work, we aim to estimate user satisfaction and dissatisfaction in conversations between users and our smartphone-based In-House virtual assistant using only the interaction patterns, without any dependency on explicit user feedback data.

Additionally, we also focus on the interpretability of negative feedback conversations where we identify reasons behind user dissatisfaction. A few examples of satisfactory (SAT) and dissatisfactory (DSAT) conversations between users and our In-House virtual assistant (In-House-VA) are illustrated in Table 1, showing domain & intent router failure and speech recognition errors. Finally, we generate similar examples of erroneous utterances for further training to handle such scenarios in the future. The research is divided into three stages.

In the first stage, our objective is to find the user dissatisfaction sessions within the system. We assess user satisfaction or dissatisfaction through interactions between users and our smartphone-based In-House-VA, involving dialog-style conversations with user requests and In-House-VA responses. We build a **Feedback Classifier** that can analyze patterns in user and In-House-VA conversations to determine whether the user experience is satisfactory or not.

After identifying negative feedback sessions, in the second stage, it is important to determine the reasons for user dissatisfaction. Therefore, we build a pipeline to identify various types of issues that can happen during user-VA interactions, we call the pipeline **Auto-Issue Categorizer**.

Smartphone-based In-House-VA data is analyzed by language experts from 2022-10-07 to 2022-10-19. The analysis focused on the top 25 smartphone domains that were most frequently used during this period. They analyzed around 8750 utterances in total, with an average of 350 frequently used utterances from each domain. By examining these utterances, five major issue types were identified, shown in Figure 1. These issue types include Automatic Speech Recognition (ASR) Failure, Multi-Intent Failure, Domain Router Failure, Intent Router Failure, and Network/Server Issues. We develop a lightweight pipeline ensemble with issue-type classifiers to address these failures.

After identifying various issue types, the next step is how to utilize this analysis and results to improve the system. The idea is simple, we generate similar erroneous samples and train the VA system to handle such scenarios in the future. We call it **Training Variant Generator**. For data generation, we leverage Mistral-7B-v0.2 (AI, 2024).

By identifying the specific issues that are causing In-House-VA to fail, developers can address these issues directly, resulting in an improved user experience. This may involve modifying In-House-VA's algorithms, language-processing capabilities, or other features to better align with user needs and preferences. Additionally, this process can provide valuable insights into how users interact with In-House-VA, allowing developers to identify areas for improvement and optimization. Ultimately, the goal of the **AsTrix** framework, shown in Figure 2, is to enhance In-House-VA's ability to meet user needs and provide more accurate, reliable and efficient performance.

2 Related Work

2.1 Literature Review

Previous studies have explored User Satisfaction Estimation (USE) through various methodologies. In 1996, (Brooke et al., 1996) introduced the System Usability Scale (SUS), a ten-item questionnaire using a 5-point Likert scale (Turner, 1993) to evaluate system ease of use and learnability. USE has been investigated in different information retrieval and natural language processing tasks, such as conversational recommender systems (Siro et al., 2023). Research has also been conducted on sentiment analysis (Song et al., 2023), content analysis (Walker et al., 1997), and response quality assessment (Schmitt and Ultes, 2015), (Bodigutla et al., 2019). Some studies have collected user satisfaction ratings using third-party annotators (van Zeelt et al., 2020). In contrast, others have employed text embeddings to estimate user satisfaction in conversational systems (Liang et al., 2021), (Kachuee et al., 2020), (Pan et al., 2022), considering both explicit and inferred user feedback (Park et al., 2020).

Additionally, transformer-based approaches have been utilized to measure user satisfaction and experience (Shen et al., 2022). Certain studies have focused on identifying dialogue acts or user intents to assess goal fulfillment (Cai and Chen, 2020), while others have proposed multi-task frameworks for user satisfaction estimation, action prediction, and utterance generation tasks (Hu et al., 2023). Research has also highlighted the importance of incorporating the sequential dynamics of dialogue acts (Deng et al., 2022), jointly predicting sentiment and satisfaction (Song et al., 2023), and modeling satisfaction dynamics across conversational turns (Ye et al., 2023).

Some works have utilized large language models (LLMs) to generate counterfactual dialogue

	Satisfactory (SAT) session			Dissatisfactory (DSAT) session		
Session ID	**Domain**	**Intent**	**User & In-House-VA Conversation**	**Domain**	**Intent**	**User & In-House-VA Conversation**
1	phone	make call	User: Call <*name*> In-House-VA: Okay; I'll call <*name*>.	phone	no action	User: Open the voicemail In-House-VA: That's not something I can help with yet.
					make call	User: Text In-House-VA: Who do you want to call?
					make call	User: Send a voice message to Sister In-House-VA: I didn't find Send a voice message to Sister in your contacts. Who should I call?
					make call	User: Sister In-House-VA: Calling Sister.
2	clock	show alarm	User: Check alarms In-House-VA: 2 turned on alarms were found. Alarm for 8 AM, Alarm for 8:40 AM every Sunday.	clock	set alarm	User: Set alarm for 5:10 AM In-House-VA: Your alarm will ring in 4 hours 55 minutes.
		disable alarm	User: Cancel the alarm for 8:40 AM In-House-VA: I turned off your alarm for 8:40 AM every Sunday.		no action	User: Saral Alam for 5:30 AM In-House-VA: I didn't understand that.

Table 1: Few examples of satisfactory (SAT) and dissatisfactory (DSAT) conversations between a user and In-House virtual assistant. The label SAT and DSAT is marked for the entire session and not on individual turns.

samples in task-oriented dialogue systems (Abolghasemi et al., 2024). (Hu et al., 2023) extended the use of LLMs as a user simulator, zero-shot prompting (Kojima et al., 2022), and few-shot prompting (Sun et al., 2023). SPUR (Lin et al., 2024) employs LLMs with supervised prompting to extract user satisfaction rubrics.

2.2 Limitations of Existing Methods

A significant amount of research focuses on using explicit user feedback data (e.g., click data and likes/dislikes), which is resource-intensive and not always readily available. Additionally, much of the existing research on User Satisfaction Estimation (USE) prioritizes classification accuracy over interpretability.

Methods based on representation learning (Song et al., 2023), (Deng et al., 2022), (Ye et al., 2023) often lack transparency due to their reliance on neural models like embeddings, providing limited insight into conversational cues that indicate satisfaction or dissatisfaction. Similarly, reward models (OpenAssistant, 2023) for training large language models (LLMs) produce continuous reward scores without explaining why certain conversations receive higher scores than others.

To address interpretability in USE, (Walker et al., 1997) evaluated user satisfaction based on human-annotated features related to task success. However, approaches relying on domain-specific features often struggle to generalize across diverse conversational patterns (Deriu et al., 2021). Research work focused solely on sentiment analysis does not fully capture USE (Song et al., 2023), and content analysis methods typically require human annotators to assess interaction quality in dialogue sessions (Schmitt and Ultes, 2015), (Bodigutla et al., 2019). LLMs with zero-shot prompting (Kojima et al., 2022) may introduce bias, as human-provided instructions may not align with actual conversation patterns in the data. Similarly, few-shot prompting (Sun et al., 2023) may not adequately represent the full distribution of conversational patterns, leading to inaccuracies in USE. (Lin et al., 2024) employs LLMs with supervised prompting for user satisfaction assessment, known as SPUR. But this approach is resource-intensive and requires pre-annotated training data for various domains. This creates challenges for multi-domain virtual assistants, making data acquisition notably difficult.

3 Proposed Methodology

This section introduces our approach in developing the integrated framework. Our comprehensive framework **AsTrix** (Assisted Triage and Fix) com-

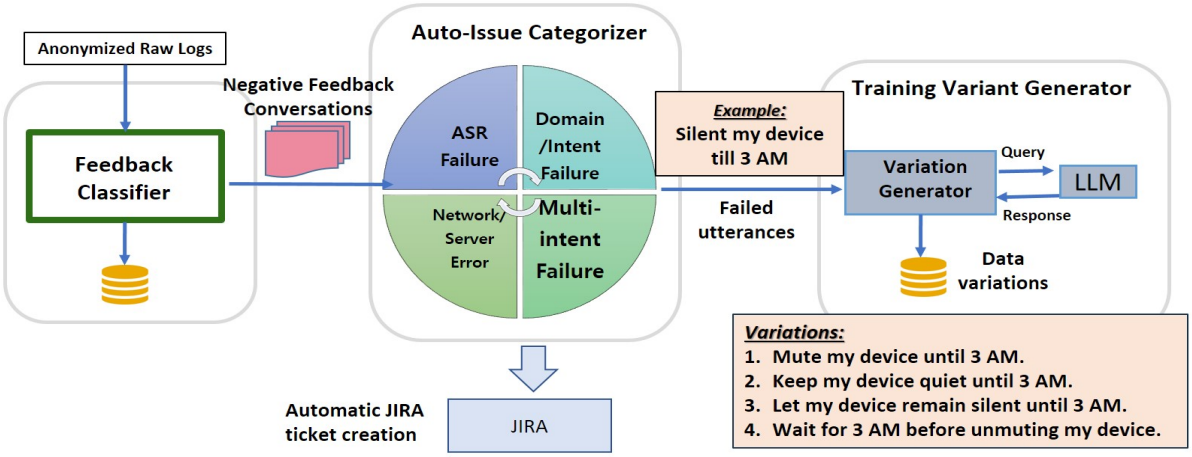

Figure 2: The AsTrix Framework

prises three distinct stages shown in Figure 2. (i) Feedback Classifier (ii) Auto-Issue Categorizer (iii) Training Variant Generator.

3.1 Stage 1: Feedback Classifier

In this stage, we construct a binary feedback classifier that labels conversations as positive or negative feedback based on user satisfaction or dissatisfaction. Considering the challenges posed by the noisy and complex raw data of large-scale In-House-VA dialogue system and also taking into account the constraints of limited computational resources, we opt to try the basic approach of unsupervised data augmentation for consistency training (UDACT)(Xie et al., 2020), which turned out to be a more effective solution, especially considering the limited labeled data and a large amount of unlabeled data.

3.1.1 Dataset

Utilizing Unsupervised Data Augmentation (UDACT) (Xie et al., 2020), we fuse a small subset of supervised training data with a larger volume of unsupervised augmented data across various In-House-VA domains (for example, phone, clock, system, gallery, music and many more). Instead of synthetic data, we leverage anonymized raw user logs to capture real conversation patterns, enabling models to better assess conversation satisfaction. In the interest of space, we cover the specifics of the datasets and data preparation details in Appendix A.1.1. The input and output details is shown under "Data Preparation" in Appendix A.1.1. The data distribution is shown in Table 6 in Appendix A.1.1. Conversations of varying lengths, ranging from single-turn to multiple turns (where a turn comprises a user utterance and In-House-VA's response), are included in the train, test, and dev dataset and are in the form of (domain, intent, user utterance, and In-House-VA response). Train, test, and validation data distribution are illustrated in Table 7 in Appendix A.1.1.

In addition to this, based on our analysis of various available datasets, we merge SNIPS[1] and MultiWOZ 2.2[2] to build a dataset with large number of domains and intents and to gain insights into model generalizability. We first split the data for supervised and unsupervised training into a 1:9 ratio as shown in Table 6 in Appendix A.1.1. Next, we transform the supervised data into positive and negative conversations based on the user's expressions of satisfaction and dissatisfaction, user sentiments, repeated misunderstanding, or rephrasing. The data distribution is shown in Table 8 in Appendix A.1.1. All details of data preparation and augmentation is illustrated in Appendix A.1.1.

3.1.2 System Architecture

The Feedback classification involves training a model using UDACT (Xie et al., 2020), that outputs positive or negative feedback, as illustrated in Figure 3. Our learning algorithm includes transfer learning on a pre-trained BERT-base model (Devlin et al., 2018). During each iteration of the training process, we calculate the supervised loss on a subset of labeled data and the consistency loss on a subset of unlabeled data. The supervised loss is cross-entropy loss, shown in eq. 1. For consis-

[1] 4https://github.com/MiuLab/SlotGated-SLU/tree/master/data/snips

[2] https://github.com/budzianowski/multiwoz/tree/master/data/MultiWOZ_2.2

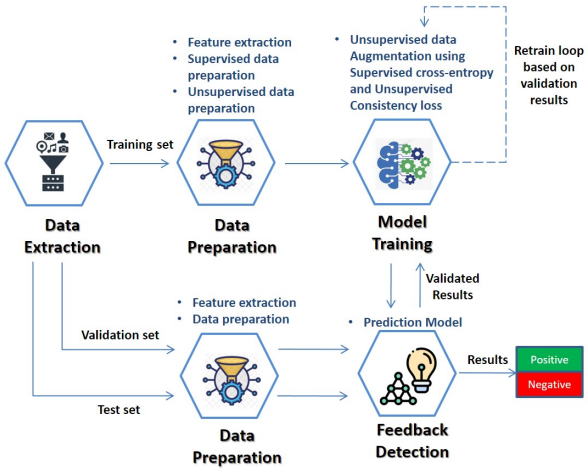

Figure 3: High Level view of Feedback Classifier

tency loss, KL Divergence between the predictions for the original input and its augmented version is used, shown in eq. 2. These two types of loss are subsequently combined to form the total loss for that iteration. The detailed process of the Feedback classification system is shown in "Algorithm 1".

$$\mathcal{L}_{\text{supervised}} = -\frac{1}{N_l}\sum_{i=1}^{N_l} y_i \log(f(x_i,\theta)) \quad (1)$$

where N_l is the number of labeled examples. y_i is the true label for the labeled example x_i. $f(x_i,\theta)$ is the predicted probability for x_i, where θ represents the model parameters.

$$\mathcal{L}_{\text{consistency}} = \frac{1}{N_u}\sum_{i=1}^{N_u} \text{KL}(f(x_i,\theta)\|f(\hat{x}_i,\theta)) \quad (2)$$

where N_u is the number of unlabeled examples. $\hat{x}_i$ is an augmented version of the unlabeled example x_i. $\text{KL}(p\|q)$ is the KL divergence between the model's predictions on the original and augmented data.

3.2 Stage 2: Auto-Issue Categorizer

3.2.1 Dataset

Out of the major five issue types (Automatic Speech Recognition (ASR) Failure, Multi-Intent Failure, Domain Router Failure, Intent Router Failure, and Network/Server Issues), we train a joint model to handle Domain Router Failure and Intent Router Failure. The remaining issue types are handled independently through zero-shot methods. The pre-processing steps of data preparation

Algorithm 1 Feedback Classification System

1: **Data Preparation:**
$S \leftarrow \text{extract\_sessions}(D)$
$S_{\text{clean}} \leftarrow \{x \in S \mid \text{is\_gibberish}(x) = 0\}$
$S_{\text{labeled}} \leftarrow \text{prepare\_labeled}(S_{\text{clean}})$
$S_{\text{unlabeled}} \leftarrow \text{prepare\_unlabeled}(S_{\text{clean}})$
2: **Training:**
$M^0 \leftarrow \text{BERT}_{\text{base}}$
for each iteration t of the training process **do**
$L^t_{\text{sup}} \leftarrow \text{supervised\_loss}(M^t, S_{\text{labeled}})$
$L^t_{\text{cons}} \leftarrow \text{consistency\_loss}(M^t, S_{\text{unlabeled}})$
$L^t_{\text{total}} \leftarrow L^t_{\text{sup}} + \lambda L^t_{\text{cons}}$
$M^{t+1} \leftarrow \text{update}(M^t, L^t_{\text{total}})$
end for
3: **Testing:**
$S_{\text{test}} \leftarrow \text{extract\_sessions}(D_{\text{test}})$
$S_{\text{test\_clean}} \leftarrow \{x \in S_{\text{test}} \mid \text{is\_gibberish}(x) = 0\}$
$Y_{\text{pred}} \leftarrow \text{predict}(M, S_{\text{test\_clean}})$
$Y_{\text{output}} \leftarrow \text{classify}(Y_{\text{pred}})$

remain the same as mentioned in section 3.1.1. Table 9 in Appendix A.1.2 shows the details of data distribution for training, testing, and validation.

3.2.2 System Architecture

To facilitate this, we design a pipeline consisting of four components:

- `Automatic Speech Recognition (ASR) Failure Detection`: There are times when In-House-VA encounters difficulties in transforming and comprehending spoken language with precision. These instances of miscommunication may stem from a variety of causes, including variations in the user's tone, accent, or pronunciation. These factors can contribute to In-House-VA's inability to properly interpret and execute the user's commands. Example shown in Table 2. To detect ASR errors in negative feedback conversations, the first utterance is compared to subsequent ones. If the utterances are of the same length with only one differing word, ASR similarity is calculated. All details are provided in Appendix A.2.1. The accuracy of ASR Failure Detection is 79.2% on 400 test samples.

- `Multi-Intent Failure Detection`: Virtual assistants sometimes fail to support multiple actions in a single user utterance, as shown in Table 2. This limitation can cause confusion

for users who expect it to execute multiple actions in response to a single command. To detect such errors, a zero-shot label-aware BERT attention network (Wu et al., 2021) is used to recognize multiple intents in a single utterance. After identifying intents, a BERT-based (Devlin et al., 2018) binary classifier checks if the virtual assistant's response aligns with the detected intents. All details are provided in Appendix A.2.1. The accuracy of Multi-Intent Failure Detection is 78% on 7050 test samples.

- `Domain and Intent Router Failure Detection`: There are cases in which In-House-VA fails to accurately determine the appropriate domain or intent based on a user's utterance. In these situations, In-House-VA may invoke actions associated with an incorrect domain or intent, resulting in unexpected or undesired behavior, as shown in Table 2. To detect such errors, a binary classifier is built using the semi-supervised method, UDACT (Xie et al., 2020), that applies data augmentation to unlabeled data and enforces prediction consistency through regularization. All details are provided in Appendix A.2.1. The accuracy of Domain and Intent Router Failure Detection is 87% on 2800 test samples.

- `Network/Server Error Detection`: At times, In-House-VA may experience difficulties in performing its functions due to network, server, or connection issues. These issues can be caused by a variety of factors, such as high traffic volumes on the server, network congestion, or connectivity problems between In-House-VA and the server. When In-House-VA is unable to connect with the server or is experiencing network issues, it may not be able to process user commands and provide appropriate responses, as shown in Table 2. These errors are detected by identifying the failure patterns in In-House-VA's response using Python's Regex library (Kuchling, 2000). All details are provided in Appendix A.2.1. The accuracy of Network/Server Error Detection is 99% on 5000 test samples.

Sequential setting: Each conversation is segmented into individual turns, where each turn is analyzed through the issue-categorizer pipeline. In this pipeline, data is handled sequentially, going through each module individually. This follows a prioritized approach, where the Automatic Speech Recognition (ASR) Failure detection module takes precedence over other modules. The reason for this is that accurate transcription is fundamental to the success of the conversation. If In-House-VA cannot accurately interpret the user's spoken commands, it is unlikely to be able to fulfill their requests, regardless of other factors that could contribute to the conversation's failure.

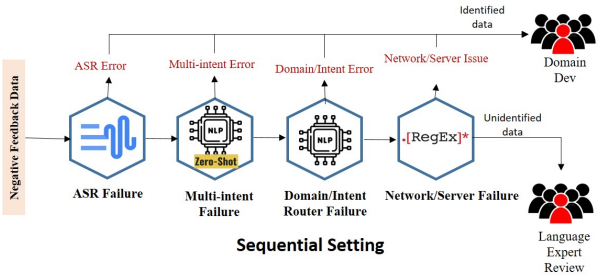

Figure 4: High Level view of Auto-issue Categorizer

After this module, the next in queue is Multi-Intent Failure detection, although multi-intent failure eventually leads to intent router failure, we opt to prioritize it because we specifically aim to identify the occurrence of failures due to multiple intents in a single user utterance. The next modules are Domain/Intent Router Failure Detection and Network/Server Error Detection module. If a sample is identified as erroneous in any module, the corresponding data is stored and the remaining data proceeds to subsequent modules. Therefore, upon reaching the end of the pipeline, the erroneous samples will be categorized under only one issue type. If there are samples that do not fall into any of the failure detection categories, then those samples are passed to the language experts for further analysis. Figure 4 shows the auto-issue categorizer pipeline.

3.3 Stage 3: Training Variant Generator

Once the erroneous instances are identified, it is crucial to fix them. In this stage, we generate variations of error utterances, which is then utilized by developers to fine-tune In-House-VA, enhancing its ability to handle similar user intents in the future. Mistral-7B-v0.2 (AI, 2024) is employed to produce these data variations. We also experiment with other less complex generative models, but they did not produce sufficient diversity in the generated data variations. As a result, we chose to use the open-source LLM, Mistral-7B-v0.2 (AI, 2024). While this LLM is not considered small and does

Issue Type	Example Conversation	Explaination
Automatic Speech Recognition (ASR) Failure	User: Put my phone on linker In-House-VA: For now; I'm not able to do that. I'll keep learning to make your experience better. User: Put my phone on ringer In-House-VA: I ran into a problem.	ASR error: ringer $\Longrightarrow$ *linker*
Multiple Intents Failure	User: In-House-VA close all apps and open WhatsApp In-House-VA: I can't help with that right now. DomainID: whatsApp IntentID: whatsApp.SendMessage	Only one intent identified
Domain Router Failure	User: Set my work alarm for 8 AM tomorrow In-House-VA: Do you want to save the event? DomainID: calendar IntentID: calendar.AddEvent	Incorrect domain (Calendar) identified instead of correct one (Clock)
Intent Router Failure	User: Hey In-House-VA send the pictures to Mumma In-House-VA: **shows gallery** DomainID: gallery IntentID: gallery.ShowContents	Incorrect intent (Show Contents) identified instead of correct one (Share Contents)
Network/Server Error	User: Restart phone In-House-VA: Should I restart it? User: ** Reboot Confirmation action ** In-House-VA: I'm having connection issues. Please try again later.	Failure due to Network/Server issue

Table 2: Few examples of various issue types identified in dissatisfactory (DSAT) sessions in smartphone-based In-House-VA

Model	In-House-VA									SNIPS-MultiWOZ2.2		
	10 domains			15 domains			30 domains			14 domains		
	F1[Positive]	F1[Negative]	Accuracy	F1[Positive]	F1[Negative]	Accuracy	F1[Positive]	F1[Negative]	Accuracy	F1[Positive]	F1[Negative]	Accuracy
Bert	90.8	90.4	90.3	89.5	83.3	87.1	86.4	80	83.9	97.2	96.8	97
FLiText	86.5	80	83.9	85.7	72.9	79.7	82.6	69.6	76.8	81.3	80.7	81
MixText	88.9	84.6	87.1	88.1	77.5	82.8	83.7	76.2	80	94.8	93.4	94.1
GPT4 - Zero Shot	80	74.1	77.4	78.6	82	80.3	74.1	78.3	76.2	97.3	96.1	96.6
GPT4 - Few Shot	68.9	72.7	71	74.2	76.5	75.4	69.2	69.7	69.5	95.1	94.5	94.8
UDA(BERT)+ Backtranslation [Ours]	93.7	93.3	93.5	91.9	90.7	91.3	90.7	90.2	90.5	98.1	96.3	97.2

Table 3: Feedback Classifier Results Compared with Baseline

come with a significant computational cost, we prioritize generating high-quality data variations. This led us to make a trade-off between computational efficiency and the quality of the output. By opting for this model, we are able to leverage its strong pre-trained capabilities, which allowed us to avoid additional fine-tuning on our specific data, while still ensuring that we meet our performance requirements.

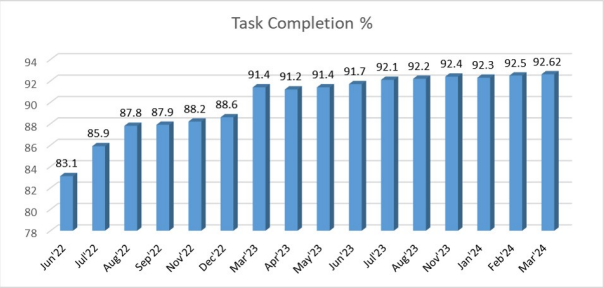

Figure 5: Increase in task completion rate of In-House-VA from Jun'22 to Mar'24

4 Experiments and Results

We perform separate experiments for the feedback classifier and the auto-issue categorizer, followed by a combined experiment using the integrated framework. Errors identified from these experiments are sent to the variation generator, where the variations are forwarded to developers for fixing the VA algorithm.

4.1 Stage 1: Feedback Classifier

4.1.1 Comparison with Baseline

For the feedback classifier, we conduct experiments with other semi-supervised methods including FLiText (Liu et al., 2021) designed for text classification tasks focusing on lightweight models like TextCNN/LSTM and MixText (Chen et al., 2020) that introduce a new data augmentation technique called TMix, which generates augmented training samples by interpolating textual data in the hidden space. We also test with GPT4 (Achiam et al., 2023) in zero-shot and few-shot settings. All experiments are conducted using a single prompting approach without any advanced techniques such as prompt tuning or chain-of-thought prompting. The reason behind opting single prompting is to evaluate the basic capabilities of large language models (LLMs) in understanding tasks when prompted with a single, straightforward input, as opposed to employing more complex strategies. Table 3 shows the comparison results on the validation set.

Discussion: Surprisingly, the results indicate that single prompting in a few-shot scenario performs worse than in a zero-shot scenario. Upon further analysis, we identify that this discrepancy is due to overfitting and the presence of bias in the

Models	In-House-VA (15 domains)			SNIPS-MWOZ2.2 (14 domains)		
	F1 [Positive]	F1 [Negative]	Accuracy	F1 [Positive]	F1 [Negative]	Accuracy
Reward Model	60.6	31.6	50	54.5	21.1	42.3
SPUR	75	60	69.2	66.7	54.4	61.5
GPT4 - Zero Shot	80	72.7	76.9	69.2	69.2	69.2
GPT4 - Few Shot	75.9	69.6	73.1	68.9	60.9	65.4
Auto-issue Categorizer [Ours]	85.7	84.2	85	79.8	79.8	79.8

Table 4: Auto-issue Categorizer Results Compared with Baseline

		Data Pre-processing	Utterance Feedback	ASR Failure	Multi-intent Failure Detection	Domain-Intent Router Failure	Network/ Server Issues	Data with No Issue Identified	Data with Issue Identified
Model Size		NA	438MB	262 MB	878MB	438MB	NA	NA	
H1'23	**Inference /Execution time**	56.54sec	6.28min	1.03min	53.38sec	54.36sec	52.17sec		
	Total Data to be identified	242744	242744	63948	63466	63055	56665	26648 (41.67%)	
	Identified		**63948**	**482**	**411**	**6390**	**30017**		**37300 (58.33%)**
H2'23	**Inference /Execution time**	1.34min	7.31min	1.21min	57.11sec	58.10sec	55.43sec		
	Total Data to be identified	259808	259808	70831	70353	69898	61577	28565 (40.32%)	
	Identified		**70831**	**478**	**455**	**8321**	**33012**		**42266 (59.67%)**
Q1'24	**Inference /Execution time**	1.03min	7.01min	1.52min	58.51sec	1.18min	59.20sec		
	Total Data to be identified	254347	254347	77673	77176	76623	68496	29413 (37.87%)	
	Identified		**77673**	**497**	**553**	**8127**	**39083**		**48260 (62.13%)**

Table 5: Improvements with Auto-issue Categorizer in smartphone-based In-House-VA

model's behavior when exposed to a small set of examples. The few-shot examples seem to guide the model too narrowly, reducing its ability to generalize effectively across the task, whereas in zero-shot settings, the model relies more on its broader, pre-trained knowledge, leading to better overall performance.

4.1.2 Improvements in In-House-VA system

Starting from June 2022, we conduct monthly monitoring of user and In-House-VA interactions. By testing two weeks of conversation data each month using the feedback classifier, we identify negative feedback conversations. We then report and developers fix these instances, resulting in a substantial increase in successful task completion rates, shown in Figure 5.

4.2 Stage 2: Auto-issue Categorizer

4.2.1 Comparison with Baseline

For auto-issue categorizer, we conduct experiments with RLHF-based reward model (OpenAssistant, 2023), LLM-based supervised prompting method SPUR (Lin et al., 2024), and single prompting with GPT4 (Achiam et al., 2023) in zero and few-shot settings. Table 4 shows the comparison results on the validation set.

Discussion: GPT-4 underperforms compared to our non-LLM ensemble method primarily because it struggles to detect certain patterns of automatic speech recognition (ASR) errors. Additionally, it often conflates the domains and intents of smartphone-based virtual assistants with more generic ones, leading to issues in recognizing specific errors or misclassifications. Furthermore, GPT-4 has difficulty distinguishing between multiple similar intents that, while distinct in the context of smartphone-based virtual assistants, are treated as synonymous by the LLM. For example, "mute volume" and "decrease volume" are considered separate intents for smartphones, but GPT-4 interprets them as having the same meaning in a generic context. This lack of differentiation contributes to its reduced performance in this domain.

4.2.2 Improvements in In-House-VA system

We evaluate the auto-issue categorizer across 1st and 2nd halves of 2023 and the 1st quarter of 2024, by analyzing 2 weeks of data per month, then averaging the results. Through continuous system

improvements, we increased the coverage of negative feedback conversations from 58% to 62%, reducing overall manual effort and enhancing system robustness. Table 5 shows the results from the auto-issue categorizer.

5 Conclusion

Our proposed framework, AsTrix, presents a streamlined approach to identify and analyze unsatisfactory interactions between users and smartphone-based virtual assistants. There could be multiple reasons behind the user dissatisfaction, and by addressing just five main issue types, we tackle around 62% of the total negative feedback conversations. This leads to a notable improvement in the success rate of user-In-House-VA interactions. While we have demonstrated the application of our framework on the smartphone-based virtual assistant, it is important to note that this method is not confined to smartphones alone. The framework is versatile and can be applied to any type of conversational agent, whether it's integrated into smart speakers, wearables, home automation systems, or customer service chatbots. Since the core focus of AsTrix is on improving the quality of interactions and addressing user dissatisfaction, it is adaptable to different platforms and devices that rely on conversational interfaces.

In the future, we aim to identify additional negative issue types, such as out-of-scope intents and slot errors, that could contribute to conversation failures. Furthermore, while retaining our non-LLM methods, we plan to harness the capabilities of Large Language Models (LLMs) to automatically identify issues based on the conversation's context. This will enable the framework to dynamically detect and categorize new or unforeseen failure types. Additionally, by utilizing advanced prompt refinement techniques like soft prompting and multi-task prompt tuning, we aim to address the limitations observed in our experiments, where single-prompting methods in zero-shot or few-shot settings with LLMs did not yield optimal results.

Limitations

In this work, we primarily focus on identifying and addressing the five major issue types contributing to failures. However, these issue types can vary across systems, and relying solely on them may not suffice to handle a significant number of failed conversations. A more effective approach would be to automatically detect and categorize new or unforeseen failure types based on the conversation's context.

We also use single prompting with LLM as a baseline to compare the results of simple, straightforward prompts with our proposed non-LLM method. However, single prompting techniques have notable limitations, including a limited ability to fully understand context, resulting in responses that lack depth or relevance. They also struggle with multi-intent queries and complex structures, as they lack mechanisms to break down or interpret different components effectively.

In future work, while maintaining the integrity and effectiveness of our non-LLM methods as a core component of the framework, we aim to address these limitations by leveraging advanced prompting techniques using LLMs. This will enable richer context understanding, improved adaptability, improved generalizability and better handling of complex queries, ultimately leading to more accurate and reliable results.

Ethics

To protect user privacy, we utilize anonymized data collected from our smartphone-based In-House Virtual Assistant. Conversations are formed by grouping query-response pairs, each identified by turnIDs, under a unique conversationID. This ensures that researchers analyzing this data cannot link the information back to any individual user or extract any information related to them.

References

Amin Abolghasemi, Zhaochun Ren, Arian Askari, Mohammad Aliannejadi, Maarten de Rijke, and Suzan Verberne. 2024. Cause: Counterfactual assessment of user satisfaction estimation in task-oriented dialogue systems. *arXiv preprint arXiv:2403.19056*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mistral AI. 2024. Mistral-7B-Instruct-v0.2. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2/. [Online; accessed 26-Feb-2024].

Praveen Kumar Bodigutla, Lazaros Polymenakos, and Spyros Matsoukas. 2019. Multi-domain conversation quality evaluation via user satisfaction estimation. *arXiv preprint arXiv:1911.08567*.

John Brooke et al. 1996. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.

Wanling Cai and Li Chen. 2020. Predicting user intents and satisfaction with dialogue-based conversational recommendations. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, pages 33–42.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. *arXiv preprint arXiv:2004.12239*.

Yang Deng, Wenxuan Zhang, Wai Lam, Hong Cheng, and Helen Meng. 2022. User satisfaction estimation with sequential dialogue act modeling in goal-oriented conversational systems. In *Proceedings of the ACM Web Conference 2022*, pages 2998–3008.

Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. 2021. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54:755–810.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ben Hixon, Eric Schneider, and Susan L Epstein. 2011. Phonemic similarity metrics to compare pronunciation methods. In *INTERSPEECH*, volume 1, pages 825–828.

Zhiyuan Hu, Yue Feng, Anh Tuan Luu, Bryan Hooi, and Aldo Lipani. 2023. Unlocking the potential of user feedback: Leveraging large language model as user simulators to enhance dialogue system. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 3953–3957.

Mohammad Kachuee, Hao Yuan, Young-Bum Kim, and Sungjin Lee. 2020. Self-supervised contrastive learning for efficient user satisfaction prediction in conversational agents. *arXiv preprint arXiv:2010.11230*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

A.M. Kuchling. 2000. Python Regex Library. https://docs.python.org/3/howto/regex.html/. [Online; accessed 1-Jan-2000].

Runze Liang, Ryuichi Takanobu, Feng-Lin Li, Ji Zhang, Haiqing Chen, and Minlie Huang. 2021. Turn-level user satisfaction estimation in e-commerce customer service. In *Proceedings of the 4th Workshop on e-Commerce and NLP*, pages 26–32.

Ying-Chun Lin, Jennifer Neville, Jack W Stokes, Longqi Yang, Tara Safavi, Mengting Wan, Scott Counts, Siddharth Suri, Reid Andersen, Xiaofeng Xu, et al. 2024. Interpretable user satisfaction estimation for conversational systems with large language models. *arXiv preprint arXiv:2403.12388*.

Chen Liu, Mengchao Zhang, Zhibin Fu, Pan Hou, and Yu Li. 2021. Flitext: a faster and lighter semi-supervised text classification with convolution networks. *arXiv preprint arXiv:2110.11869*.

Moushumi Mahato, Avinash Kumar, V Kiran Kumar Reddy, and Javaid Nabi. 2023. A unified framework for detecting domain and intent misclassifications in large-scale dialogue systems. In *2023 IEEE Women in Technology Conference (WINTECHCON)*, pages 1–6. IEEE.

OpenAssistant. 2023. Reward Model Trained from Human Feedback. https://huggingface.co/OpenAssistant/reward-model-deberta-v3-large-v2/. [Online; accessed 26-Jan-2023].

Yan Pan, Mingyang Ma, Bernhard Pflugfelder, and Georg Groh. 2022. User satisfaction modeling with domain adaptation in task-oriented dialogue systems. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 630–636.

Dookun Park, Hao Yuan, Dongmin Kim, Yinglei Zhang, Matsoukas Spyros, Young-Bum Kim, Ruhi Sarikaya, Edward Guo, Yuan Ling, Kevin Quinn, et al. 2020. Large-scale hybrid approach for predicting user satisfaction with conversational agents. *arXiv preprint arXiv:2006.07113*.

Hieu Pham, Xinyi Wang, Yiming Yang, and Graham Neubig. 2021. Meta back-translation. *arXiv preprint arXiv:2102.07847*.

Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588.

Alexander Schmitt and Stefan Ultes. 2015. Interaction quality: assessing the quality of ongoing spoken dialog interaction by experts—and how it relates to user satisfaction. *Speech Communication*, 74:12–36.

Wei Shen, Xiaonan He, Chuheng Zhang, Xuyun Zhang, and Jian Xie. 2022. A transformer-based user satisfaction prediction for proactive interaction mechanism in dueros. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1777–1786.

Clemencia Siro, Mohammad Aliannejadi, and Maarten De Rijke. 2023. Understanding and predicting user satisfaction with conversational recommender systems. *ACM Transactions on Information Systems*, 42(2):1–37.

Kaisong Song, Yangyang Kang, Jiawei Liu, Xurui Li, Changlong Sun, and Xiaozhong Liu. 2023. A speaker turn-aware multi-task adversarial network for joint user satisfaction estimation and sentiment analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13582–13590.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. *arXiv preprint arXiv:2305.08377*.

Jean Turner. 1993. Using likert scales in l2 research. another researcher comments. *TESOL quarterly*, 27(4):736–739.

Naushad UzZaman and Mumit Khan. 2005. A double metaphone encoding for bangla and its application in spelling checker. In *2005 international conference on natural language processing and knowledge engineering*, pages 705–710. IEEE.

Mickey van Zeelt, Floris den Hengst, and Seyyed Hadi Hashemi. 2020. Collecting high-quality dialogue user satisfaction ratings with third-party annotators. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, pages 363–367.

Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. *arXiv preprint cmp-lg/9704004*.

Ting-Wei Wu, Ruolin Su, and Biing Juang. 2021. A label-aware bert attention network for zero-shot multi-intent detection in spoken language understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4884–4896.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in neural information processing systems*, 33:6256–6268.

Fanghua Ye, Zhiyuan Hu, and Emine Yilmaz. 2023. Modeling user satisfaction dynamics in dialogue via hawkes process. *arXiv preprint arXiv:2305.12594*.

A Appendix

A.1 Dataset details

A.1.1 Stage 1: Feedback Classifier

As shown in Table 6, for feedback classifier, we experiment with mostly used top 10, 15, and 30 smartphone domains of In-House-VA.

Dataset	Domain	Intent	Super-vised	Unsuper-vised
In-House-VA	10	143	7091	2658048
	15	184	10101	3112492
	30	331	19397	4518223
SNIPS-MultiWOZ2.2	14	37	9892	247925

Table 6: Supervised and Unsupervised Data Distribution of In-House-VA and SNIPS-MultiWOZ2.2

Train, test, and validation data distribution of In-House-VA and SNIPS-MultiWOZ2.2 for feedback classification is illustrated in Table 7 and Table 8 respectively. The positive data comprises conversations wherein tasks are successfully executed, indicating user satisfaction, labeled as 'positive', while the negative data comprises those where tasks encounter failure, indicating user dissatisfaction, labeled as 'negative'.

Domain	In-House-VA			
		Total	Positive	Negative
10	**Train**	7091	3549	3542
	Test	2081	1040	1041
	Dev	2006	1002	1004
15	**Train**	10101	5060	5041
	Test	4521	2264	2257
	Dev	3375	1873	1502
30	**Train**	19397	9822	9575
	Test	7633	3982	3651
	Dev	6113	3078	3035

Table 7: In-House-VA Supervised Data Distribution

Domain	SNIPS-MultiWOZ2.2			
		Total	Positive	Negative
14	**Train**	9892	4946	4946
	Test	2775	1384	1391
	Dev	2553	1280	1273

Table 8: SNIPS-MultiWOZ2.2 Supervised Data Distribution

Data Preparation details include:

- `Data Cleaning`: As the smartphone-based In-House-VA dataset is sourced directly from raw logs, it would inevitably include nonsensical samples. We use a pre-built gibberish classifier to filter out the gibberish or meaningless samples. This gibberish classifier is a binary classifier and is trained on the In-House-VA labeled dataset. This classifier outputs two labels, 'gibberish' and 'non-gibberish'. Currently, the gibberish classifier provides an accuracy of 86.7%. We do not discuss how to implement this classifier in this paper as we solely focus on evaluating the tasks performed by virtual assistants.

- `Data Preparation`: For data samples, we transform every turn of a conversation in the form ⟨DomainID⟩@ ⟨IntentID⟩@ ⟨Utterance⟩@ ⟨In-House-VA response⟩, separated by *[SEP]*. To address conflicting intents, we combine intents that are highly similar.

  (a) For supervised data preparation, we prepare data with two labels, label Positive signifies 'Positive Feedback' and label Negative signifies 'Negative Feedback' of the conversation. For label Positive, we take conversations in which tasks are successfully executed by In-House-VA, for label Negative, we take conversations in which tasks are not successfully executed.

  As illustrated in Table 1, an example of a Positive session is: *phone @ phone make call @ Call ⟨name⟩@ Okay; I'll call ⟨name⟩.*

  Similarly, an example of a Negative session is: *phone @ phone no action @ Open the voicemail @ That's not something I can help with yet. [SEP] phone @ phone make call @ Text @ Who do you want to call? [SEP] phone @ phone make call @ Send a voice message to Sister @ I didn't find Send a voice message to Sister in your contacts. Who should I call? [SEP] phone @ phone make call @ Sister @ Calling Sister.*

  (b) For unsupervised augmented data preparation, we prepare two sub-samples for each sample, we transform every utterance into an original and augmented utterance in the form ⟨DomainID⟩@ ⟨IntentID⟩@ ⟨Original Utterance⟩@ ⟨In-House-VA response⟩and ⟨DomainID⟩@ ⟨IntentID⟩@ ⟨Augmented Utterance⟩@ ⟨In-House-VA response⟩.

  An example of the original text is: *clock @ clock set alarm @ hey make alarm after 1 hour @ Okay, alarm will ring after 1 hour.*

An example of augmented text is: *clock @ clock set alarm @ alert me after an hour @ Okay, alarm will ring after 1 hour.*

- `Data Augmentation with Back-translation`: We augment all data using the back-translation (Pham et al., 2021) method, a process of translating a target language text back into its source language by using a machine translation system. We translate English sentences into French and then revert them back to English to generate text augmentations.

A.1.2 Stage 2: Auto-issue Categorizer

Train, test and dev data distribution of the auto-issue categorizer is shown in Table 9. Only Domain and Intent Router Failure module is trained, remaining modules are zero-shot methods.

Dataset	Domain	Intent	Train	Test	Dev
			Domain and Intent Router Failure		
In-House-VA	15	184	9367	8038	5231
SNIPS-MultiWOZ-2.2	14	37	5643	2013	2005

Table 9: Auto-issue Categorizer Data Distribution

A.2 System Architecture details

A.2.1 Stage 2: Auto-issue categorizer

The Auto-issue Categorizer pipeline consists of four components:

- `Automatic Speech Recognition (ASR) Failure Detection`: To detect ASR errors in negative feedback conversations, the first utterance is compared with the subsequent utterances in that conversation. If the pair of utterances is of the same length and all words are same except one, the common words are removed and the uncommon words are compared for ASR similarity. ASR similarity is calculated using two methods: (1) Phoneme-based word similarity (Hixon et al., 2011): The phoneme representation of two words is extracted, and their similarity is determined using Jaccard distance. A threshold of 0.5 classifies utterances as potential ASR errors. (2) Double metaphone-based distance (UzZaman and Khan, 2005): The metaphone representation of two words is extracted, and the Jaccard distance is used to calculate the metaphone distance. Utterances are classified as potential ASR errors if the distance is less than or equal to 1.

- `Multi-Intent Failure Detection`: To address this challenge, a solution is formed by using a zero-shot label-aware BERT attention network (Wu et al., 2021). The model is capable of recognizing multiple intents in a single utterance without prior knowledge of the intent labels. Post recognizing multiple intents, their successful execution as tasks is detected by checking the relevance between the intents and response by VA through a BERT-based binary classifier.

  The approach involves two key steps: (1) Attention Mechanism: A transformer-based BERT model (Devlin et al., 2018) encodes input utterances and intent labels, which are then processed through a label-aware attention mechanism (Wu et al., 2021) that assigns weights to words based on their relevance to the intent. This allows the model to focus on intent-relevant words. (2) Multi-Intent Classification: The attention output is concatenated with the encoded utterance and passed through a multi-layer perceptron (MLP) (Popescu et al., 2009), which maps the input to a probability distribution, predicting the intent with the highest probability. (3) Relevance Calculation: Using a BERT-based (Devlin et al., 2018) binary classifier, the relevance between identified intents and virtual assistant response is measured to ensure the task is successfully completed by the virtual assistant.

- `Domain and Intent Router Failure Detection`: Inspired by (Mahato et al., 2023), the solution involves building a binary classifier using UDACT (Xie et al., 2020), a semi-supervised learning method. UDACT (Xie et al., 2020) enhances training by applying data augmentation to unlabeled data and enforcing consistency between model predictions on original and augmented data through regularization. It combines supervised learning on a small set of labeled data with unsupervised learning to improve generalization and accuracy.

- `Network/Server Error Detection`: We find such issues using Python's Regex library (Kuchling, 2000) that helps to identify patterns of network, server, or connection failure in In-House-VA's response.