

Article

NLU Difficulty-Aware Routing for Conversational Expert Systems, Avoiding Post-Hoc Cascading Through Pre-Inference Decision Making

Subhin Yang, Donghwan Kim and Sungju Lee *

Department of Software, Sangmyung University, Cheonan 31066, Republic of Korea;
2024D4005@sangmyung.kr (S.Y.); 2024D1009@sangmyung.kr (D.K.)

* Correspondence: peacfeel@smu.ac.kr

Abstract

Hybrid approaches for conversational NLU commonly rely on cascading strategies, where model selection is performed after inference based on output confidence or uncertainty. Although effective, such post-hoc decision processes introduce redundant computation and accumulated latency due to sequential model execution. In this paper, we propose a pre-inference routing framework that selects an appropriate model before executing any candidate NLU model and without relying on candidate-model confidence or uncertainty. Instead of relying on output-dependent signals, we model NLU difficulty as a multi-dimensional representation of the joint intent recognition and slot extraction task, thereby distinguishing the proposed approach from prior post-hoc routing methods that rely primarily on confidence or uncertainty. To further reflect real-world dialogue characteristics, we incorporate temporal conversational factors, including contextual dependency and topic transitions. Based on these features, a lightweight routing mechanism efficiently assigns each utterance to an appropriate model from a pool of domain-specialized and general-purpose models. Experimental results on single-domain turns show that the proposed method maintains competitive Semantic Accuracy while reducing high-capacity model invocation, cost, and latency, thereby improving inference efficiency. These results demonstrate that input-observable difficulty features can provide a practical basis for pre-inference model selection without relying on post-hoc cascading.

Keywords: natural language understanding; pre-inference routing; NLU difficulty; hybrid NLU; model routing; intent recognition; conversational systems; cascading

Academic Editor: George A. Tsihrintzis

Received: 26 June 2026

Revised: 30 July 2026

Accepted: 31 July 2026

Published: 2 August 2026

Copyright: © 2026 by the authors.
Licensee MDPI, Basel, Switzerland.
This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Efficient and accurate Natural Language Understanding (NLU) is essential for deploying real-time task-oriented conversational systems. In such systems, NLU modules must reliably identify user intents and extract the slot values required to complete tasks across multi-turn dialogues [1,2]. However, real-world user utterances exhibit varying levels of difficulty due to factors such as ambiguity, and multi-turn contextual dependencies [3]. Some utterances can be effectively processed by lightweight rules or small language models, consistent with prior resource-constrained task-oriented NLU systems that employ rule-based components to reduce processing overhead [4]. In contrast, more

complex utterances demand stronger semantic reasoning to handle multiple constraints and complex dialogue dynamics [5]. Managing this variability without compromising computational efficiency remains a critical challenge for practical systems [6].

Recent advances in Large Language Models (LLMs) have significantly improved the robustness of conversational NLU, particularly for these complex and context-dependent cases. However, applying LLMs to all utterances is computationally expensive and introduces substantial latency, making them impractical for real-time systems [7]. In practical conversational services, uniformly applying a high-capacity model to every utterance can result in unnecessary resource consumption. Commercial language-model APIs may differ in inference cost by up to two orders of magnitude [8]. The practical importance of input-dependent processing is also reflected in industrial conversational NLU. Among 2550 unique user utterances collected from live logs of Walmart's e-commerce voice assistant, 40.11% required inter-utterance context for correct intent interpretation. In the same study, a BERT-based NLU implementation required approximately 450 ms per query on a CPU, whereas a lightweight GloVe-based implementation required approximately 5 ms [9]. These findings illustrate that real conversational workloads contain inputs with heterogeneous processing requirements and that model choice can substantially affect inference latency. To address these efficiency and heterogeneity challenges, hybrid NLU architectures have been proposed that combine lightweight models with high-capacity models and selectively invoke more expensive models when necessary [7,10–12]. Most existing hybrid NLU architectures rely on cascading, where a smaller model first processes the utterance and defers to a larger model when its confidence is low or its uncertainty is high [13]. Although cascading can reduce average inference cost relative to LLM-only systems, it introduces structural limitations. Post-hoc decision-making, such as cascading, inherently requires at least one model inference and often leads to redundant computation and accumulated latency.

In contrast, we argue that model selection should be performed prior to inference, based solely on the characteristics of the input utterance. To address this challenge, we propose a NLU difficulty-aware routing framework that estimates utterance complexity directly from input-level features.

Unlike prior work that focuses on output uncertainty or response quality, our approach models the intrinsic difficulty of NLU tasks. Specifically, we decompose difficulty into intent ambiguity, reflecting uncertainty in user goals, and slot-space ambiguity, representing the difficulty of extracting structured information. These components are further augmented with temporal conversational features, including contextual dependency, topic transitions, and topic return patterns, which capture the dynamics of multi-turn dialogue. Based on these features, a lightweight routing strategy determines the most suitable model from a model pool before inference. In addition, the predicted model level is then combined with domain information to dispatch the utterance to an appropriate model in a domain-specific expert pool. The experimental results demonstrate that the proposed method successfully selects an appropriate model level before inference, achieving a Semantic Accuracy comparable to that of LLM-only and cascading-based methods while significantly reducing LLM invocations, computational cost, and latency. Comparisons with multiclass and embedding-based pre-inference routers also showed that the proposed method maintained competitive Semantic Accuracy while reducing Total Latency relative to the alternative routers.

The main contributions of this paper are summarized as follows:

1. We propose a pre-inference routing framework that eliminates cascading by performing model selection prior to execution.

2. We introduce an interpretable utterance complexity representation that captures task-level ambiguity, including intent ambiguity, slot-space ambiguity, and transaction risk, together with temporal conversational cues and surface-level signals.
3. We formulate NLU difficulty as an explicit multi-dimensional representation of the joint intent recognition and slot extraction task and demonstrate that it can support appropriate model selection before inference, reducing computational redundancy compared with post-hoc cascading while maintaining competitive Semantic Accuracy.

2. Related Work

2.1. Hybrid NLU and Model Routing

To balance performance and efficiency in conversational systems, recent studies have explored hybrid architectures that combine lightweight models with large LLMs [7,10,11,14]. These approaches aim to reduce computational cost by selectively invoking high-capacity models only when necessary. In particular, cascading-based frameworks have been widely adopted, where a lightweight model first processes the input and defers uncertain or complex cases to more powerful models [5,10,11].

Several studies have investigated uncertainty-driven routing strategies. For example, softmax-based confidence methods use the maximum probability of the final classification layer as a reliability signal for accepting or rejecting the prediction [10]. Similarly, uncertainty-based routing methods estimate prediction uncertainty using techniques such as Monte Carlo dropout and use the uncertainty score to decide whether escalation to an LLM is required [11]. Recent work has also extended cascading to generative language models. Language model cascade frameworks define deferral policies based on token-level uncertainty, allowing the system to continue generation with a larger model when the smaller model becomes uncertain [13]. In dialogue-oriented hybrid systems, Context-Bridge combines rule-based processing with LLM-based fallback conditions to improve intent and dialogue understanding in IVSR scenarios [7]. These studies show that hybrid routing can reduce unnecessary LLM usage compared with LLM-only inference.

However, most of these methods rely on post-hoc output signals, such as confidence, uncertainty, or failure conditions. Therefore, the routing decision can only be made after at least one model has already been executed. As a result, cascading inherently introduces redundant computation and accumulated latency when escalation to higher-capacity models is required [6].

2.2. Intent Recognition and Slot Filling

Intent recognition is a fundamental task in task-oriented dialogue systems, aiming to identify the user's goal from an utterance [15,16]. In practical task-oriented NLU, intent recognition is closely coupled with slot filling, which extracts structured information required to complete the task. Modern approaches therefore commonly adopt joint intent classification and slot filling models. For example, BERT-based joint models learn shared contextual representations for intent classification and slot tagging [17], while Transformer- and graph-based models further capture interactions between multiple intents and slot labels [18–20]. These studies demonstrate that jointly modeling intent and slot information is more suitable for structured NLU than treating the two subtasks independently. In real-world applications, successful understanding requires both intent and slot predictions to be correct, as reflected in metrics such as Semantic Accuracy [20,21]. This observation suggests that the difficulty of NLU tasks cannot be fully captured by considering either component independently, motivating the need for joint difficulty modeling.

2.3. Multi-Turn Dialogue and Contextual Understanding

In multi-turn dialogue settings, understanding user utterances requires modeling contextual dependencies across turns. Prior work has incorporated dialogue history using techniques such as sliding window context tracking, cross-encoder representations, and retrieval-based methods [22]. These approaches improve intent prediction by capturing semantic similarity and contextual relevance between utterances.

However, these existing approaches primarily focus on enhancing representation learning for prediction tasks, rather than explicitly modeling how contextual factors affect the difficulty of NLU. In particular, conversational phenomena such as deixis, topic shifts, and topic returns introduce ambiguity that cannot be resolved from a single utterance alone.

2.4. Limitations of Existing Model Routing Strategies

Although prior studies have made significant progress in hybrid routing and conversational NLU, they exhibit several limitations. First, most routing strategies rely on output-dependent signals such as confidence or uncertainty, which require at least one model inference and lead to additional computational overhead. Second, cascading-based frameworks introduce sequential dependencies between models, resulting in accumulated latency and redundant inference cost. Third, existing approaches do not explicitly model the intrinsic difficulty of NLU tasks, particularly in terms of the joint difficulty of intent recognition and slot filling. Finally, while contextual information is often used to improve prediction performance, its role in determining routing decisions remains underexplored.

To address these limitations, we propose an NLU difficulty-aware routing framework that performs model selection prior to inference. Unlike existing methods, our approach estimates utterance complexity directly from input-level features, without relying on their output confidence or uncertainty. Furthermore, we introduce a fine-grained difficulty modeling scheme that jointly considers intent ambiguity and slot-space ambiguity, and incorporates temporal conversational features to capture multi-turn dynamics. Table 1 presents a comparison of existing hybrid routing, Dialogue NLU routing, and LLM routing approaches and the proposed method. Most hybrid routing methods [7,10,11,13] are based on cascading, in which model selection is performed after inference using output-dependent signals such as confidence or uncertainty. As a result, these methods incur accumulated cost and latency whenever escalation to a higher-capacity model is required. Although context-aware dialogue models [5] can exploit pre-inference information, their routing signals are typically based on dense representations, such as embedding similarity. Such cues are useful for context modeling or prediction support, but they do not explicitly capture the task-level complexity of NLU, including intent ambiguity, slot-space ambiguity, and dialogue-state-dependent semantic constraints. Recent studies have also investigated cost-aware routing and model selection for LLM-based systems. FrugalGPT [8] reduces inference cost by selecting or cascading among multiple LLMs, while RouteLLM [23] learns to route queries between weaker and stronger LLMs based on expected response quality and cost. However, these studies mainly focus on LLM-to-LLM routing for general-purpose generation tasks. In contrast, our work addresses difficulty-aware routing for task-oriented dialogue NLU, where real-world user utterances exhibit different levels of intent ambiguity, slot complexity, and contextual dependency. More specifically, unlike FrugalGPT, which relies on post-hoc cascading among LLMs, and RouteLLM, which does not explicitly model intent- and slot-level difficulty, the proposed framework performs pre-inference routing across heterogeneous model levels, including LLMs, using NLU-specific difficulty signals. The proposed router estimates NLU task difficulty before executing candidate models and assigns utterances to appropriate levels within a heterogeneous pool of rule-based components, fine-tuned encoders, and an LLM

under a Semantic Accuracy criterion. To support this decision, it explicitly models task-specific NLU difficulty using intent ambiguity, slot-space ambiguity, transaction risk, contextual and temporal dependencies, and surface-level characteristics. These dimensions are more directly interpretable than embedding-based representations, allowing the routing rationale to be examined in terms of identifiable NLU difficulty factors.

Table 1. Comparison of Conversational NLU Routing Approaches and the Proposed Method.

Category	Approach	Decision Signal	Decision Timing	Multi-Turn Context	Cascading	Key Limitation
Hybrid Routing	Cascading-based [10]	Softmax Confidence Score	Post-inference	No	Yes	Redundant computation, latency accumulation
	Uncertainty-based routing (MC Dropout, Softmax) [11]	Uncertainty	Post-inference	No	Yes	Requires model execution for decision
	LM Cascade [13]	Token-level Uncertainty	Post-inference	No	Yes	Sequential dependency and inference overhead
	ContextBridge [7]	Rule + failure condition	Post-inference	Partial	Yes	Rule-dependent context handling
Dialogue Systems	Context-aware Models [5]	Embedding Similarity	Pre-inference	Yes	No	No fine-grained difficulty modeling
LLM Routing	FrugalGPT [8]	Output quality scoring	Post-inference	No	Yes	Cascading overhead, No NLU-specific difficulty modeling
	RouteLLM [23]	Learned preference/quality-cost predictor	Pre-inference	No	No	Query-level routing only, No intent/slot-level difficulty modeling
Ours	Proposed Method	NLU difficulty (Intent + Slot + Context)	Pre-inference	Yes	No	Depends on feature quality of complexity representation

2.5. Utterance Difficulty Estimation for NLU

Estimating the difficulty of an input utterance is important for efficient model selection in task-oriented NLU. Existing approaches commonly rely on model confidence, uncertainty, or embedding-based similarity to identify difficult inputs. Confidence- and uncertainty-based methods can indicate whether a model is likely to fail, but they require the model to be executed before the difficulty signal becomes available [7,10,11,13]. Embedding-based routing methods can estimate semantic distance or distributional novelty before inference [5,23], but they often represent difficulty through a single similarity-based signal.

However, utterance difficulty in task-oriented NLU is difficult to determine by semantic distance alone. It is also affected by task-specific ambiguity, such as intent ambiguity and slot-space ambiguity, as well as multi-turn conversational phenomena, including deixis, contextual similarity, and topic return [14,24–26]. These factors are particularly important because NLU requires joint intent and slot prediction under dialogue context. Despite this, relatively few studies have directly quantified task-level difficulty with respect to intent and slot prediction, and utterance difficulty in this context has largely been inferred indirectly through downstream signals such as model confidence or embedding similarity, rather than through task-specific NLU difficulty factors.

Therefore, this study adopts an eight-dimensional input-observable difficulty representation that combines task-level, contextual, and surface-level signals. Unlike approaches that approximate difficulty using a single dominant signal, the proposed representation explicitly fuses intent-related, slot-related, lexical, and dialogue-history features before executing candidate NLU models.

3. Pre-Inference NLU Difficulty-Aware Routing

Conventional cascading performs routing only after a preceding model has already been executed. As a result, the routing decision depends on post-hoc model outputs and may introduce redundant inference when escalation is required. In contrast, the proposed framework makes the routing decision before any NLU model execution. Given a user utterance and its dialogue context, the framework extracts input-observable complexity signals and predicts the model level required for joint intent and slot prediction.

In this study, we present a pre-inference NLU-aware routing approach for conversational expert systems. The proposed method aims to efficiently select an appropriate model for each user utterance prior to model execution, thereby avoiding the limitations of cascading-based approaches. Specifically, we estimate the complexity of an utterance directly from input-level features, without executing candidate NLU models or relying on their output confidence or uncertainty. The estimated complexity reflects the joint difficulty of intent recognition and slot filling, as well as contextual dependencies in multi-turn dialogue. Based on this complexity, a routing decision is made to assign the utterance to a suitable model from a heterogeneous model pool. This design enables efficient and accurate natural language understanding while reducing computational cost and latency in real-time conversational systems.

The proposed method consists of three components: (1) a multi-dimensional NLU difficulty estimation feature that quantifies utterance complexity from input-level features prior to model execution; (2) a lightweight router that maps the estimated complexity vector to an appropriate model level; and (3) a heterogeneous model pool spanning capability levels. We define NLU difficulty as the likelihood of incorrect joint prediction of intent and slots under lightweight inference conditions. Figure 1 shows that the proposed framework estimates NLU difficulty directly from the input utterance and dialogue context using lightweight features, including intent ambiguity, slot-space ambiguity, and temporal conversational characteristics. Based on the estimated difficulty, a routing decision is made prior to model inference, selecting the most appropriate model from a heterogeneous model pool.

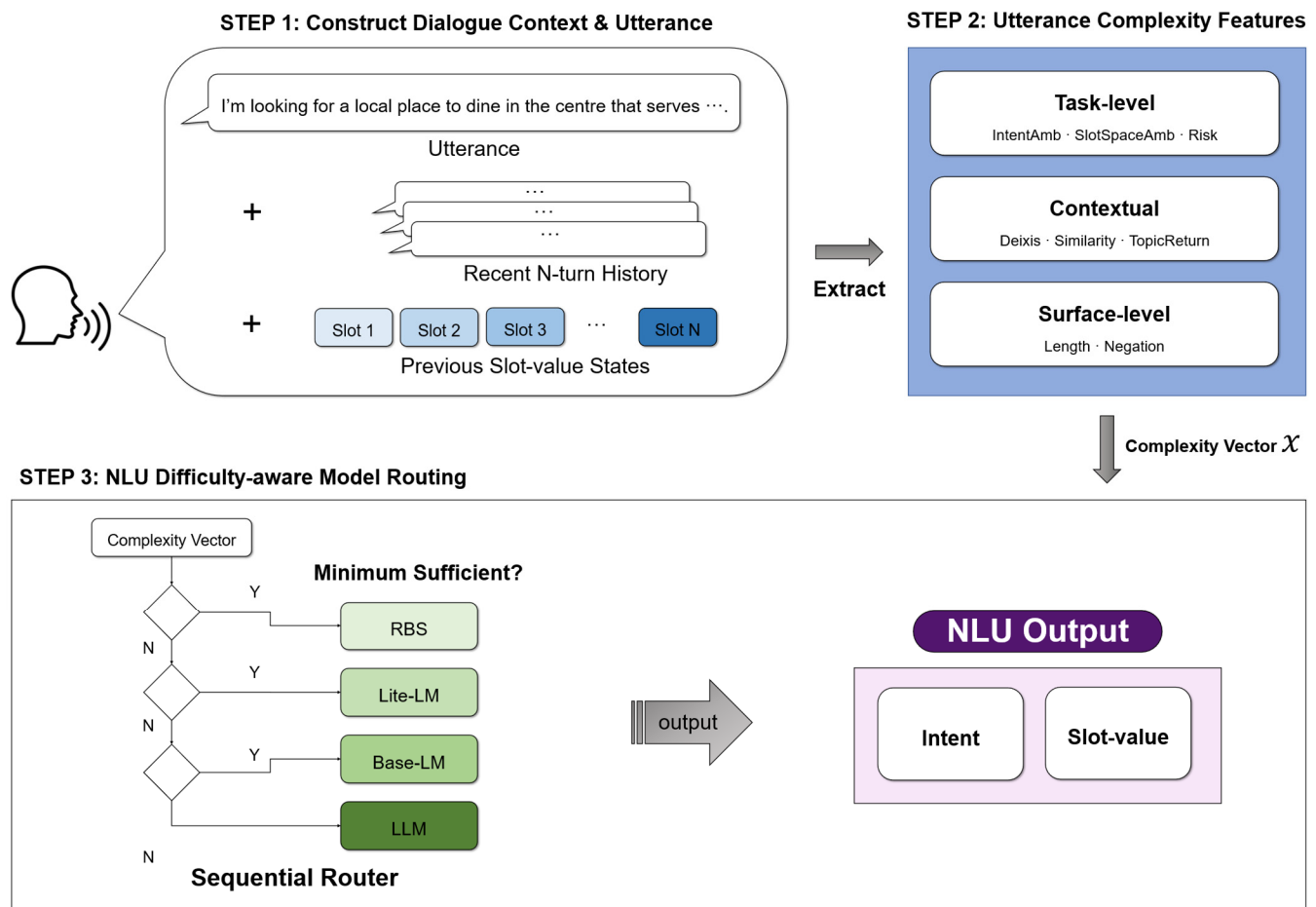


Figure 1. Overview of the Proposed NLU Difficulty-aware Pre-inference Routing.

3.1. Pre-Inference NLU Difficulty Representation

Since the routing decision must be made before model execution, we approximate this complexity using input-observable signals extracted from the utterance and its dialogue context. This representation serves as the evidence available to the router in the pre-inference setting, replacing output-dependent signals such as confidence or uncertainty used in cascading methods.

The proposed representation consists of three groups of signals. Task-level complexity captures the semantic burden directly associated with intent and slot prediction. Contextual complexity captures dependency on dialogue history and non-local conversational flow. Surface-level indicators provide additional lightweight lexical cues directly observable from the utterance. Table 2 summarizes the features used in each group.

Table 2. Definition of Utterance Complexity Features for Model-level Routing.

Complexity	Description	Feature (Alias)
Task-level	Task-level signals reflecting the difficulty of the NLU task, where a larger set of candidate intents and slots increases the difficulty of correctly identifying and extracting structured information.	Intent Ambiguity (IntentAmb) SlotSpace Ambiguity (SlotSpaceAmb) Transaction Risk (Risk)
Contextual	A measure of the degree to which the understanding of the current utterance relies on dialogue history beyond the utterance itself.	Deixis (Deixis) Topic Return (TopicReturn) Contextual Similarity (Sim)
Surface-level	Surface-level signals capture lexical and syntactic difficulty signals that are directly observable from the current utterance, without requiring dialogue history.	Utterance Length (Len) Negation (Neg)

We employ eight input-level features to construct a unified representation of NLU difficulty. Given a user utterance u , its domain d , and dialogue context c , the routing input vector is defined by concatenating task-level, contextual, and surface-level feature vectors as follows:

$$\begin{aligned} \mathbf{X} &= [\mathbf{x}_{\text{Task}}; \mathbf{x}_{\text{Contextual}}; \mathbf{x}_{\text{Surface}}] \in \mathbb{R}^8, \\ &\text{where} \\ \mathbf{x}_{\text{Task}} &= (\text{IntentAmb}(u), \text{SlotSpaceAmb}(u, d), \text{Risk}(u)), \\ \mathbf{x}_{\text{Contextual}} &= (\text{Deixis}(u), \text{Sim}(u, c), \text{TopicReturn}(u, c)), \\ \mathbf{x}_{\text{Surface}} &= (\text{Len}(u), \text{Neg}(u)) \end{aligned} \quad (1)$$

The resulting feature vector $\mathbf{X}$ is used as the input to the routing model:

$$\hat{l} = f_{\text{router}}(\mathbf{x}) \quad (2)$$

where $\hat{l}$ denotes the predicted model level for utterance u .

3.1.1. Task-Level Complexity

Task-level complexity captures task-related factors that affect the reliability and consequence of NLU predictions. In task-oriented dialogue, the system must identify the user's intent and extract the slot values required to execute the corresponding task. Intent ambiguity occurs when an utterance contains cues associated with multiple plausible intents, whereas slot-space ambiguity increases when several slot values or constraints must be identified and assigned correctly within a single utterance. In addition, some utterances involve transaction-related operations, such as booking, changing, or cancelling a reservation. Errors in these operations may directly alter the dialogue state or lead to an unintended system action, whereas errors in information-seeking requests are more likely to result in an incorrect recommendation or response. Accordingly, we represent task-level complexity using three signals: intent ambiguity, slot-space ambiguity, and transaction risk.

Intent Ambiguity (IntentAmb)

Ambiguity captures the degree to which surface-level lexical cues in an utterance activate multiple intent candidates within a domain. Even when the domain is correctly identified, multiple intent candidates may exist within that domain, increasing the difficulty of intent classification. For example, in the hotel domain, an utterance such as "I need a hotel" clearly corresponds to a single intent (find_hotel), whereas "yes, I am looking for a hotel in a moderate price range with 2 stars and I will need to book 5 people for 4 nights" contains lexical cues associated with both 'find_hotel' and 'book_hotel'. In this case, the utterance partially shares surface-level expressions with the find_hotel intent while simultaneously including booking-related cues resulting in the activation of multiple intent candidates. Such simultaneous activation makes intent classification more difficult because the model must distinguish between competing intent labels that share similar lexical evidence [12,27].

To measure this surface-level intent ambiguity, the utterance is matched against a keyword-based schema to identify intent candidates within each domain. For each domain d , the number of activated intent candidates is counted, and the largest candidate count across domains is used to compute the Intent Ambiguity as follows:

$$\text{IntentAmb}(u) = \begin{cases} 0 & \text{if } M_{\text{intra}}(u) \leq 1 \\ 1 - \frac{1}{M_{\text{intra}}(u)} & \text{if } M_{\text{intra}}(u) \geq 2 \end{cases} \quad (3)$$

The M_{intra} is then defined as:

$$M_{\text{intra}}(u) = \max_{d \in D} |\text{Intents}_d(u)| \quad (4)$$

where D is the set of all domains. This formulation assigns zero ambiguity when a single intent is clearly identified and increases as the number of competing intent candidates grows within a domain. Since the main experiments are conducted under an oracle-domain setting, this feature focuses on intra-domain ambiguity rather than inter-domain ambiguity.

Slot Space Ambiguity (SlotSpaceAmb)

Slot Space Ambiguity (SlotSpaceAmb) captures the difficulty of extracting multiple slot values that are simultaneously expressed within a single utterance. In task-oriented dialogue, users often provide several constraints at once, requiring the system to identify and correctly extract multiple slot-value pairs in a single step. As the number of such constraints increases, the extraction task becomes more challenging due to the increased risk of omission, misalignment, or incorrect assignment [28,29]. For example, the utterance “I need a cheap Italian restaurant in the centre and book it for 4 people at 19:00 on Friday” contains multiple slot constraints, including price range, food, area, number of people, time, and day. In such cases, multiple slot values are activated simultaneously from surface-level lexical cues, increasing the complexity of accurate slot extraction.

To quantify this complexity, we measure the proportion of detected slot values relative to the total number of possible slots within a domain. Specifically, Slot Space Ambiguity is defined as:

$$\text{SlotSpaceAmb}(u, d) = \frac{|\text{DetectedSlots}(u, d)|}{|\text{TotalSlots}(d)|} \quad (5)$$

where $\text{DetectedSlots}(u, d)$ denotes the set of slot values identified in utterance u , and $|\text{TotalSlots}(d)|$ is the total number of slots defined for the domain. A higher value indicates that more domain-relevant slot values are activated in the current utterance, increasing the burden of producing an exact slot-value set. While IntentAmb reflects ambiguity in selecting the correct intent, SlotSpaceAmb captures the burden of extracting the complete set of structured slot values. Since Semantic Accuracy requires both the intent and all slot values to be correct, these two features provide complementary task-level evidence for pre-inference model selection.

Transaction Risk (Risk)

Transaction Risk identifies utterances that involve state-changing or execution-sensitive operations, such as booking, changing, or cancelling a reservation. Unlike information-seeking requests, for which an incorrect interpretation generally results in an inaccurate response or recommendation, transaction-related requests may lead to an incorrect reservation, cancellation, or modification of the dialogue state [14]. Therefore, even when such utterances do not exhibit strong intent or slot ambiguity, they should be treated as error-sensitive cases in model routing.

We detect transaction-related cues using a predefined lexicon including expressions such as book, reserve, change, and cancel. Risk is represented as a binary indicator:

$$\text{Risk}(u) = \begin{cases} 1, & \text{if } u \text{ contains transaction – related cues} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

Together, these task-level features represent the difficulty directly associated with the current NLU task, including intent identification, slot-value extraction, and transaction-sensitive interpretation. However, in multi-turn conversational systems, accurately understanding a user’s goal often requires looking beyond a single turn. To address these inter-turn dependencies, we introduce contextual complexity.

3.1.2. Contextual Complexity

Contextual complexity captures the extent to which the current utterance depends on preceding dialogue context for correct interpretation. Prior studies on multi-turn dialogue understanding have shown that conversational data is often difficult to process due to contextual phenomena such as referential expressions, implicit mentions, and topic transitions [14,25,26]. Motivated by this observation, we represent contextual complexity using three complementary pre-inference signals: Deixis, Contextual Similarity, and Topic Return.

Deixis (Deixis)

Deixis captures explicit referential expressions whose interpretation requires prior context. For example, in “Yes, that will work. I need to book it for 2 people,” the expressions “that” and “it” cannot be resolved from the current utterance alone. We therefore represent Deixis as a binary indicator based on the occurrence of predefined referential expressions, such as “this,” “that,” and “it.”

Contextual Similarity (Sim)

Contextual Similarity (Sim) measures semantic continuity between the current utterance and recent dialogue history. Building on prior studies showing that dialogue history provides contextual evidence for identifying user intents and slot values [24], we compute contextual similarity to estimate the extent to which the current utterance depends on past conversational context. It is used to identify whether the current utterance remains close to the ongoing conversational context or introduces content that is less related to recent turns. Unlike Deixis, which captures explicit referential expressions, similarity provides a continuous signal of contextual continuity even when no explicit reference term appears.

Given the current utterance u_t and the dialogue context c_t available at turn t , Sim measures the maximum cosine similarity between u_t and the preceding N user turns. Although c_t includes both previous utterances and the accumulated dialogue state, this feature uses only the recent utterance history for similarity computation.

$$\text{Sim}(u_t, c_t) = \max_{i \in [t-N, t-1]} \text{cosine}(f(u_t), f(u_i)) \quad (7)$$

where u_i denotes a previous utterance within the preceding N turns and $f(\cdot)$ denotes the TF-IDF vector representation of an utterance. A high Sim value indicates that the current utterance is semantically close to at least one recent turn, whereas a low value indicates weak continuity with the recent dialogue context.

Topic Return (TopicReturn)

While Sim captures general semantic continuity with recent dialogue history, it is challenging to discern whether the current utterance continues the immediately preceding topic or returns to an earlier topic after an intervening shift. To capture this non-local dependency, we define Topic Return.

Topic Return captures cases in which the current utterance is more closely related to an earlier dialogue turn than to the immediately preceding turn. This pattern occurs when a dialogue temporarily shifts to another task or domain and later resumes a previously discussed task. TopicReturn is defined as:

$$\text{TopicReturn}(u_t, c_t) = \max \left(0, \max_{i \in [t-N, t-2]} \text{Sim}(u_t, u_i) - \text{Sim}(u_t, u_{t-1}) \right) \quad (8)$$

where u_{t-1} denotes the immediately preceding utterance and u_i denotes an earlier utterance within the preceding N turns. TopicReturn was computed using cosine similarity between TF-IDF representations of the current and preceding utterances. A high TopicReturn value indicates that the current utterance is more strongly associated with an earlier conversational context than with the immediately preceding turn. For example, when a

dialogue first concerns hotel selection, then moves to attraction information, and later returns to modifying the hotel booking, the final utterance may require the system to recover an earlier task state rather than continue from the most recent topic.

Together, Deixis, Contextual Similarity, and TopicReturn represent complementary aspects of contextual dependency. Deixis captures explicit reference to prior context, Contextual Similarity captures general continuity or change across recent turns, and TopicReturn captures non-local resumption of a previously discussed task. These features allow the router to consider dialogue-history dependency before selecting a model level.

3.1.3. Surface-Level Complexity

Surface-level complexity captures lexical and syntactic characteristics of an utterance, such as sentence length and the presence of negation, which may increase parsing difficulty and lead to errors in understanding, even when the underlying intent is relatively simple. In other words, surface complexity reflects complexity signals that are directly observable from the current utterance, without requiring dialogue history.

Utterance Length (Len)

Utterance Length (Len) is included as a lightweight surface cue because longer utterances often contain more constraints, modifiers, or slot-value expressions, thereby increasing the amount of information that the NLU model must process. Prior studies have also used utterance length as a lexical feature for intent classification, suggesting that length provides a simple but informative signal for utterance-level understanding [5]. In this study, Len is computed as the normalized token count of the current utterance.

Negation (Neg)

Negation (Neg) indicates whether the utterance contains an explicit negative expression. Since negation has been identified as a challenge in many natural language understanding tasks [21], we include it as a surface-level complexity cue. In slot-oriented NLU, negation is relevant because a mentioned value may correspond to an excluded condition rather than a requested constraint. For example, in “I do not want a guesthouse,” the value “guesthouse” appears in the utterance but should not be assigned as the user’s preferred accommodation type. We represent Neg as a binary indicator based on a predefined lexicon of negative expressions, such as “not,” “don’t,” and “can’t.”

These surface-level indicators serve as lightweight supplementary cues that complement task-level and contextual features. Together, the three feature groups constitute a unified NLU difficulty representation, which provides the pre-inference evidence used by the model-level router described in Section 3.2.

3.2. Difficulty-Aware Model-Level Routing

Based on the estimated complexity of each user utterance, the proposed method selects an appropriate model from a predefined model pool prior to NLU model execution. Given the utterance complexity feature vector x defined in Section 3.1, routing is formulated as a pre-inference decision problem based on utterance-level NLU difficulty. The objective is not to select the most capable model, but to select the minimum model level that is sufficient for producing a semantically correct NLU output. As shown in Equation (2), the router takes the utterance complexity feature vector x as input and predicts the required model level l .

3.2.1. Routing Label Construction

We train a lightweight router that maps the utterance complexity feature vector to a model-level routing decision. To train the router, each utterance is assigned a target

model-level label. The target label represents the minimum model level, under the predefined capacity ordering, that is sufficient to produce a semantically correct NLU output. A prediction is considered semantically correct only when both the intent label and the complete slot-value set match the ground truth.

For an utterance u_i with dialogue context c_i , the correctness of the model at level l is represented by the instance-level success indicator $s_l(u, c)$ defined as:

$$s_l(u_i, c_i) = \mathbb{I} \left[(\widehat{y}_{i,l}^I = y_i^I) \wedge (\widehat{y}_{i,l}^S = y_i^S) \right] \quad (9)$$

y_i^I and y_i^S denote the ground-truth intent label and slot-value set for utterance i and $\widehat{y}_{i,l}^I$ and $\widehat{y}_{i,l}^S$ denote the intent and slot predictions produced by model level l .

The target routing label is then defined as:

$$l_i^* = \min \{ l \in L \mid s_l(u_i, c_i) = 1 \} \quad (10)$$

where $L = \{l_0, l_1, \dots, l_k\}$ is an ordered model pool from lower-capacity to higher-capacity models. This capacity ordering represents the computational hierarchy of the model pool, rather than an assumption of monotonic improvement in Semantic Accuracy across model levels. If multiple model levels correctly process the same utterance, the lowest successful level is selected as the target label. This labeling strategy prevents unnecessary assignment to higher-capacity models while preserving the semantic correctness requirement.

If none of the candidate models produces a semantically correct output, the instance is recorded as Unresolved for diagnostic analysis and mapped to the highest model level, LLM, for target-label construction and routing evaluation. Because the Unresolved status is not observable at inference time, these instances are routed to the LLM by default, and any remaining LLM failures are retained as incorrect cases when computing Semantic Accuracy.

3.2.2. Router Training and Inference

The router is trained to predict the target model-level label from the eight-dimensional utterance complexity feature vector. Because the routing decision is performed before NLU inference, the router should introduce minimal computational overhead. It should also provide interpretable decision rules so that the relationship between utterance complexity features and model selection can be analyzed. For this reason, we use a decision tree as the routing model.

At inference time, the system first extracts utterance complexity feature vector x from the current utterance and dialogue context. The trained router then predicts l , and the utterance is forwarded to the model corresponding to the predicted model level. Since the routing decision is completed before any NLU model is executed, the proposed method avoids the sequential trial-and-escalation procedure used in cascading approaches.

After the router predicts the required model level, the selected level is combined with the domain label to dispatch the utterance to the corresponding domain-specific model instance. In the present experiments, this domain label is provided by the ground-truth annotation of MultiWOZ 2.2 under an oracle-domain setting. Accordingly, the reported results isolate model-level routing performance and do not evaluate the accuracy or cost of an end-to-end domain detection module.

The proposed router adopts a sequential architecture in which multiple Decision Tree gates are connected in series. The model pool is arranged in ascending order of computational capacity, and the router traverses this hierarchy to identify the minimum sufficient model level capable of handling a given utterance. Beginning with the most lightweight model, each gate evaluates whether the current model is sufficient for the utterance at hand; upon the first gate whose sufficiency condition is satisfied, the utterance is

immediately routed to the corresponding model. Utterances that do not satisfy any lower-level gate are ultimately assigned to the highest-capacity model, i.e., the LLM.

Unlike a single multiclass Decision Tree that predicts the target model in a single pass, the proposed sequential architecture explicitly encodes the ordinal structure of the model pool by decomposing the routing decision into a series of binary sufficiency gates. Furthermore, the white-box nature of Decision Trees enables post-hoc analysis of which utterance complexity features contributed most significantly to the routing decision at each gate, providing a high degree of interpretability into the model selection process. Figure 2 illustrates the overall inference pipeline of the proposed Sequential Router. At each decision gate, the Yes branch indicates that the current model is deemed sufficient, routing the utterance to that model. Conversely, the No branch cascades the utterance to the next, higher-capacity model for further evaluation.

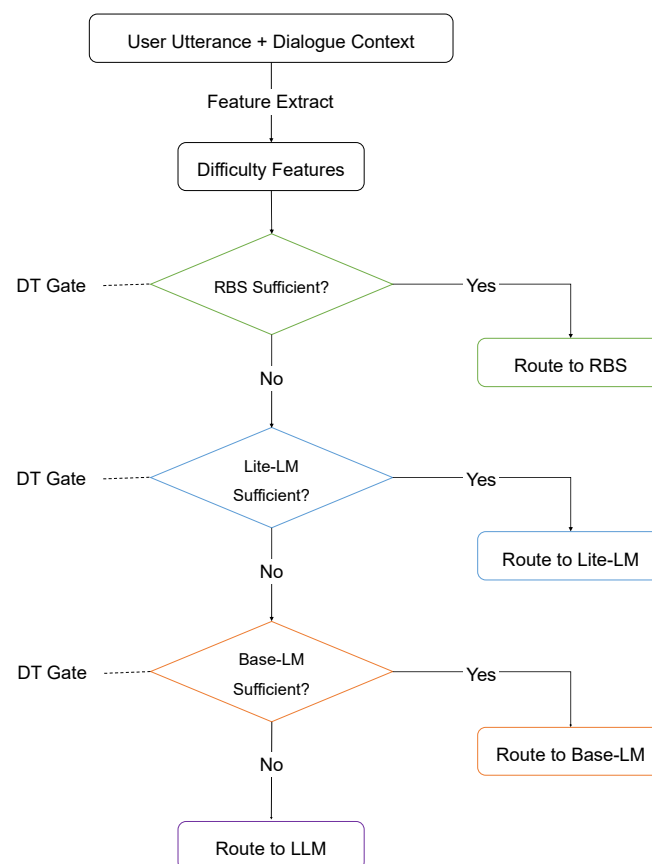


Figure 2. Sequential Decision Tree Router for Minimum Sufficient Model Selection.

4. Experiments

4.1. Experiments Setup

4.1.1. Dataset

We evaluate the proposed framework on MultiWOZ 2.2 [30], a multi-domain task-oriented dialogue benchmark. The dataset contains multi-turn dialogues across eight domains, including hotel, restaurant, train, taxi, attraction, hospital, police, and bus. We use the official training, validation, and test splits.

MultiWOZ 2.2 is suitable for this study because it provides multi-turn dialogue contexts and annotated intent-slot structures, allowing us to evaluate both NLU correctness and routing decisions under contextual dependency. In particular, the dataset includes utterances involving topic shifts, referential expressions, and multiple slot constraints, which are related to the proposed utterance complexity features.

To construct domain-specific expert model pools, we used only single-domain turns in the main experiments. In addition, domains with fewer than 1000 training samples, including bus, police, and hospital, were excluded from the main domain-wise evaluation to reduce the instability caused by small sample sizes. As a result, the main evaluation was conducted on six domains: Hotel, Restaurant, Train, Taxi, Attraction, and No Domain. In all training, validation, and test experiments, the ground-truth domain labels provided by MultiWOZ 2.2 were used. Accordingly, the experiments were conducted under an oracle-domain setting to isolate model-level routing performance from errors introduced by domain prediction. The No Domain category was also assigned based on the original dataset annotations. Although the target turn was single-domain, dialogue history was preserved for contextual feature extraction, allowing TopicReturn to capture returns to earlier entities, constraints, or task contexts after intervening turns.

4.1.2. Feature Extraction

The eight complexity features were extracted using predefined schema information, manually defined lexical rules, and dialogue-history signals. IntentAmb was computed using intent-related keywords derived from the service and slot descriptions in the predefined schema. SlotSpaceAmb was calculated from domain-specific slot values identified through regular-expression patterns, manually defined aliases, and value-normalization rules. Lexical variants, such as “inexpensive” to “cheap,” “center” to “centre,” and “guest house” to “guesthouse,” were normalized.

Risk, Deixis, and Neg were represented as binary indicators based on predefined lexical cue sets. Len was computed as the number of word and numeric tokens extracted using a regular-expression tokenizer. The rule sets used for slot cue matching, intent cue matching, transaction-risk detection, deixis detection, and negation detection are provided in Appendix A. Sim and TopicReturn were calculated using TF-IDF-based cosine similarity between the current utterance and preceding dialogue turns, following the definitions in Section 3.1. Unmatched expressions did not activate the corresponding lexical features, and no test-set annotations were used to construct the feature-extraction resources.

4.1.3. Implementation Environment

All experiments were conducted using Python 3.10.19. The local NLU models and routing modules were implemented with PyTorch 2.10.0, HuggingFace Transformers 4.57.6, and scikit-learn 1.6.1. The experiments were performed on a workstation equipped with an AMD Ryzen 5 5600X 6-Core Processor, 16 GB RAM, and one NVIDIA GeForce RTX 3060 GPU with 12 GB VRAM. The GPU environment used CUDA 12.2 runtime support and CUDA 11.8 toolkit support. The LLM-based inference was conducted using the OpenAI Python SDK 2.26.0. The local NLU models were fine-tuned using the NVIDIA GeForce RTX 3060 GPU. However, to evaluate inference efficiency under a lightweight deployment setting, all latency measurements, including feature extraction, routing decision, and local model inference, were conducted on the CPU. This common execution environment was used to examine whether pre-inference routing can reduce latency relative to post-hoc routing without reducing Semantic Accuracy, while minimizing the influence of heterogeneous hardware resources on the comparison.

4.1.4. Model Pool

The model pool consists of multiple NLU models with different capacity and cost levels. Each model receives the current utterance, dialogue history, and previous dialogue state as input, and produces intent and slot predictions. The router is trained to select the

minimum sufficient model level based on the difficulty representation defined in Section 3.1.

Table 3 presents the configuration of the model pool. In this study, we employ four representative models with different capacity and cost characteristics: a rule-based system based on predefined domain rules, DistilBERT [31] as a distilled and lightweight variant of BERT, RoBERTa [32] as a robustly optimized pretrained Language Model (LM), and GPT-5-mini (gpt-5-mini-2025-08-07) as a cost-efficient lightweight LLM. The LLM was queried using the Chat Completions API with model gpt-5-mini-2025-08-07. The temperature and top_p parameters used their respective default values of 1.0, and no explicit maximum output-token limit was specified in the API request. The generated outputs were parsed using a rule-based line parser that extracts the intent: line and comma-separated slot=value pairs from the slots: line. The complete prompt is provided in Appendix B. The model pool is organized into four ordered levels, ranging from a rule-based system to a LLM. If none of the models in the pool produces a semantically correct output, the corresponding turn is treated as an unresolved case. Each model is provided with the recent three dialogue rounds as contextual input. The Lite-LM and Base-LM are fine-tuned for domain-specific NLU using a multi-head architecture, whereas the LLM is used through prompt engineering without additional fine-tuning. The LLM was conditioned on the domain-specific slot ontology, valid intent set, output schema, and eight in-context examples, as shown in Appendix B. To provide a supplementary comparison of model usage, we assigned predefined normalized weights of 0.01, 0.20, 0.40, and 1.00 to RBS, Lite-LM, Base-LM, and LLM, respectively, according to their ordered computational levels. For the auxiliary routing components, normalized overhead weights of 0.001, 0.003, 0.002, 0.001, and 0.05 were assigned to handcrafted feature extraction, Sequential DT routing, single-stage DT routing, MLP routing, and sentence-embedding extraction, respectively. These weights do not represent measured monetary costs or a calibrated hardware-cost model. Because the present study focuses on pre-inference routing rather than cost-model estimation, the resulting metric is used only as a supplementary indicator, while measured latency and LLM Call Rate constitute the primary efficiency measures. For readability, this normalized relative-cost indicator is hereafter referred to simply as Cost. For static and proposed routing methods, the cost is computed based on the selected model. For Confidence Cascade, the cost is accumulated over all executed models along the escalation path.

Table 3. Configuration of Model Pool.

Level	Model	Specification	Role	Cost
0	RBS	-	Rule-based System	0.01
1	Lite-LM	DistilBERT-base [31]	Lightweight Model	0.2
2	Base-LM	RoBERTa-base [32]	Middle-capacity Model	0.4
3	LLM	GPT-5-mini (gpt-5-mini-2025-08-07)	Highest-capacity Model	1.0

To evaluate the efficiency of the proposed pre-inference NLU difficulty-aware routing method, we applied Confidence Cascade to the same model pool and compared it with the proposed method. In Confidence Cascade, a lower-level model is executed first, and a confidence score is obtained by applying softmax to the logits of the final layer. If the confidence score is lower than a predefined threshold, the utterance is escalated to the next higher-level model. We also compared the proposed method with a multi-model parallel ensemble baseline. Prior studies have explored various ensemble strategies, including boosting-based intent classification and neural ensemble models that combine multiple base models through an MLP-based aggregation framework [14,33,34]. Following this

direction, the parallel ensemble baseline executes all models in the model pool and uses their prediction results and confidence scores as input to an MLP-based meta-classifier, which selects the model output to be used as the final NLU prediction. The MLP consisted of two hidden layers with 32 and 16 units, using ReLU activation and a softmax output layer. It was trained with Adam, a learning rate of 0.001, a batch size of auto, and up to 500 iterations with early stopping using the development-set performance for model selection.

To compare the proposed sequential pre-inference router with alternative pre-inference structures, we additionally evaluated three baselines: a single-stage multiclass DT (MultiClass DT) using the same handcrafted difficulty features, an embedding-based DT router (Embedding DT), and an embedding-based MLP router (Embedding MLP). All methods were evaluated under the same evaluation protocol. The embedding-based routers used ‘sentence-transformers/all-MiniLM-L6-v2’, a Sentence-BERT-based model from the Sentence Transformers library. The DT baselines followed the same training and hyperparameter-selection settings as the proposed DT router, while the embedding-based MLP used the same layer configuration as the MLP employed in the Parallel Ensemble.

For each domain, the Minimum Sufficient Success label was used as the training label for the Decision Tree router. The labels were generated from the inference results of the model pool on the training set of MultiWOZ 2.2. To avoid data leakage, we applied 5-fold cross-validation and constructed out-of-fold prediction files. Based on Semantic Accuracy, each turn was assigned the lowest model level that produced a semantically correct output. The domain-wise distribution of the Minimum Sufficient Success labels is shown in Table 4. The label set consists of five categories: RBS, Lite-LM, Base-LM, LLM, and Unresolved. For the No Domain category, which mainly consists of greetings, thanks, and other non-task-bearing expressions, most turns can be handled by low-resource models such as RBS or Lite-LM. In contrast, domains with large slot spaces, such as Hotel, show a relatively higher proportion of unresolved cases, suggesting that these domains contain turns that are difficult to handle within the current model pool. The corresponding Unresolved rates in the final test set were 25.23%, 13.48%, 26.05%, 0.00%, 14.57%, and 0.00% for Hotel, Restaurant, Train, Taxi, Attraction, and No Domain, respectively, with an overall rate of 16.20%.

Table 4. Distribution of Minimum Sufficient Models by Domain.

Domain	RBS	Lite-LM	Base-LM	LLM	Unresolved	Total
Hotel	381	2879	1134	4653	5656	14,703
Restaurant	2685	4582	875	3067	3699	14,908
Train	280	329	184	7073	4122	11,988
Taxi	487	1674	3	629	622	3415
Attraction	638	1868	1003	2039	2493	8041
No Domain	3088	5791	0	0	0	8879

The Decision Tree routers were optimized by grid search over hyperparameters, including the maximum tree depth and the minimum number of samples per leaf. The grid search considered max_depth values of [2, 3, 4, 5, 6] and min_samples_leaf values of [10, 20, 50, 100]. We also searched min_samples_split values of [20, 50, 100], with class_weight = “balanced”. In the sequential router, each gate was formulated as a binary sufficiency classifier and trained on the remaining samples that were not assigned to lower model levels. Unresolved cases were not introduced as an additional routing class; instead, they were retained as negative instances for the lower-level gates, and instances rejected by the final gate were routed to the LLM by default. To mitigate class imbalance in gate-level success prediction, balanced class weights were applied during training. At inference

time, the positive-class probability obtained from ‘predict_proba’ was used as an estimated sufficiency score. The gate-specific decision thresholds were selected on the development set based on the Semantic Accuracy–latency Pareto frontier. For each domain and routing method, all candidate threshold configurations were evaluated, and the resulting objectives were normalized to identify the Pareto knee, which served as a common reference for defining the minimum acceptable Semantic Accuracy. Pareto-optimal configurations with Semantic Accuracy below that of the knee point were excluded. The final threshold combination was then determined from the remaining non-dominated, accuracy-preserving region of the Pareto frontier, favoring higher Semantic Accuracy while avoiding a disproportionate increase in total latency. The selected threshold configuration was then fixed and evaluated on the test set. The threshold-selection procedure was applied to both the Confidence Cascade and the proposed routing method. The final thresholds used for the proposed router and the Confidence Cascade are reported in Appendix C.

4.2. Metrics

We evaluate the proposed framework from two perspectives: NLU prediction quality and routing efficiency. NLU prediction quality evaluates intent and slot prediction correctness, while routing efficiency evaluates whether the router reduces high-capacity model invocation without sacrificing semantic correctness.

4.2.1. NLU Performance Metrics

For NLU quality, we use Semantic Accuracy as the primary metric because the routing label is constructed based on the joint correctness of intent and slot predictions. A prediction is considered semantically correct only when both the intent label and the complete slot-value set match the ground truth. Semantic Accuracy is defined as:

$$\text{Semantic Accuracy} = \frac{1}{N} \sum_{i=1}^N 1[(\widehat{y}_i^I = y_i^I) \wedge (\widehat{y}_i^S = y_i^S)] \quad (11)$$

where y_i^I and $\widehat{y}_i^I$ denote the ground-truth and predicted intent labels, and y_i^S and $\widehat{y}_i^S$ denote the ground-truth and predicted slot-value sets for utterance i .

We additionally report Intent Accuracy and Slot F1 to analyze subtask-level performance. Intent Accuracy measures whether the predicted intent matches the ground truth intent, while Slot F1 measures the precision-recall trade-off of predicted slot-value pairs. Intent Accuracy and Semantic Accuracy are computed as utterance-level correctness rates over the complete test set, whereas Slot F1 is micro-averaged across all test instances. These metrics are used as diagnostic indicators, whereas Semantic Accuracy is used as the main criterion for model-level labeling and overall NLU evaluation.

4.2.2. Routing Efficiency and Risk

For routing evaluation, we measure how effectively the router selects the minimum sufficient model level before inference. Under-routing occurs when the router selects a model level lower than the target level, whereas Over-routing occurs when the router selects a higher model level than necessary:

$$\text{UnderRouting} = \frac{1}{N} \sum_{i=1}^N 1[\widehat{l}_i < l_i^*] \quad (12)$$

$$\text{OverRouting} = \frac{1}{N} \sum_{i=1}^N 1[\widehat{l}_i > l_i^*] \quad (13)$$

UnderRouting is treated as a more critical routing error because it may reduce NLU correctness by assigning an utterance to an insufficient model. In contrast, OverRouting mainly affects efficiency by invoking a higher-capacity model than necessary. Because

Unresolved instances have no minimum sufficient model level within the candidate pool, they were excluded from the denominators of the UnderRouting and OverRouting rates. However, these instances were retained in the overall NLU evaluation, and unsuccessful final predictions were counted as incorrect when computing Semantic Accuracy.

To evaluate efficiency, we report the LLM Call Rate as the high-capacity model invocation rate, average inference cost, and average latency. The high-capacity model invocation rate measures the proportion of utterances routed to the highest-capacity model, such as an LLM. Cost and Total Latency were computed by averaging the predefined cost and measured latency associated with the model selected for each utterance. Latency was measured with a batch size of 1 and averaged over all instances in the test set. For the proposed method, Total Latency was calculated as the sum of utterance-complexity feature extraction, pre-inference routing, and inference by the selected model. Therefore, we decompose the total latency into three components: feature extraction latency, routing decision latency, and model inference latency. Since the LLM was accessed through an external API, its latency was measured as client-side end-to-end wall-clock latency. Accordingly, LLM latency included API request transmission, network round-trip time, server-side processing, and response generation, while local prompt construction and output parsing were excluded.

4.3. Overall Performance and Efficiency Comparison

To evaluate the NLU performance and inference efficiency of the proposed pre-inference NLU difficulty-aware routing method, experiments were conducted across six domains. Table 5 presents a method-wise overall performance comparison on the test set. In addition, we include Ideal as an upper-bound reference that represents the best possible model selection for each utterance, assuming prior knowledge of the ground-truth routing label. Although Ideal is not a deployable routing method, it provides an upper-bound reference for the maximum achievable performance of the routing framework under ideal model selection. We assume that all candidate models are executed simultaneously for each input in a parallel ensemble setting. Therefore, the computational cost accumulates across the entire model pool, whereas the inference latency is determined by the slowest model under true parallel execution. Among the static routing methods, the three non-LLMs, namely RBS, Lite-LM, and Base-LM, achieved low latency. However, because all utterances were processed by a single fixed model, their Semantic Accuracy (SemAcc) was limited to 0.28, 0.61, and 0.68, respectively. In particular, although Lite-LM and Base-LM achieved high Intent Accuracy (IntAcc), their Slot F1 scores were not sufficient to maintain high SemAcc. This indicates that accurate intent prediction alone is insufficient for task-oriented NLU, where both intent and slot-value predictions must be correct. In contrast, the LLM-based static method achieved the highest prediction performance among the static methods; however, it incurred inference latency of 4217.57 ms. For dynamic routing, the proposed method routed only 18% of the utterances to the LLM while achieving a SemAcc of 0.74, which is higher than the few-shot prompted LLM-only baseline with a SemAcc of 0.69 and comparable to the Confidence Cascade with a SemAcc of 0.73. Parallel Ensemble attained the highest SemAcc of 0.83 by exhaustively executing all candidate models. However, this gain was achieved at the cost of substantially higher normalized cost and latency. When compared with alternative pre-inference routing approaches, the proposed method achieved higher SemAcc than MultiClass DT and Embedding DT and recorded the lowest Total Latency among the four pre-inference routers. It also provided Semantic Accuracy comparable to that of Embedding MLP, which achieved the highest SemAcc. These results indicate that the objective of the proposed method is not to maximize Semantic Accuracy through exhaustive model execution, but rather to provide a

deployable, efficiency-oriented trade-off that reduces LLM invocation, cost, and latency while maintaining competitive NLU performance.

Table 5. Overall Performance Comparison by Method.

Routing	Method	SemAcc	IntAcc	Slot F1	LLM Call Rate	Cost	Total Latency
Static	RBS	0.28	0.38	0.24	-	0.01	0.10
	Lite-LM	0.61	0.98	0.70	-	0.20	50.92
	Base-LM	0.68	0.98	0.79	-	0.40	100.72
	LLM	0.69	0.92	0.84	1.00	1.00	4217.57
Dynamic (Post-hoc)	Confidence	0.73	0.89	0.68	0.24	0.49	1127.10
	Cascade						
	Parallel Ensemble	0.83	0.97	0.91	1.00	1.61	4217.85
Dynamic (Pre-inference)	MultiClass DT	0.64	0.88	0.78	0.23	0.35	1135.72
	Embedding DT	0.68	0.91	0.82	0.23	0.36	1077.11
	Embedding MLP	0.75	0.94	0.86	0.31	0.46	1452.64
	Ideal	0.88	0.94	0.91	0.33	0.38	874.37
	Proposed	0.74	0.96	0.85	0.18	0.35	806.75

To verify whether pre-inference routing based on utterance complexity is effective across domains, we constructed domain-specific expert model pools and evaluated the proposed method in each domain. Table 6 presents the detailed domain-wise NLU performance results for analyzing the effectiveness of routing under different levels of domain complexity. As shown in Table 6, in complex domains with a large number of slots and high contextual dependency, such as Hotel, Train, Restaurant, and Attraction, the lower-level static models maintained high IntAcc, whereas their Slot F1 scores decreased substantially, resulting in lower Semantic Accuracy (SemAcc). This indicates that domains with more challenging slot extraction require more flexible model selection, because intent-level correctness alone does not guarantee complete semantic correctness. In contrast, Taxi and No Domain represent relatively simple domains, where the evaluated turns mostly involve intent-level decisions with no slot values to extract. Therefore, Slot F1 is not reported for these domains. The Taxi domain shows relatively low Semantic Accuracy for the few-shot prompted LLM-only baseline. This may be due to its small and structurally constrained output space, where local encoder-based classifiers can directly model predefined labels, while the generative LLM may produce plausible but schema-inconsistent outputs. In such low-complexity domains, lower-capacity models such as RBS and Lite-LM achieved SemAcc scores close to 1.00. The proposed method captures these domain-specific characteristics before inference, thereby avoiding unnecessary high-level model invocation in simple domains while assigning more difficult utterances in complex domains to appropriate higher-capacity models. Consequently, the proposed method provides NLU performance comparable to the few-shot prompted LLM-only baseline and Confidence Cascade, while avoiding unnecessary high-capacity model invocation across domains.

Table 6. NLU Performance Comparison by Domain and Method.

Domain	Number of Test Turns	Method	SemAcc	IntAcc	Slot F1
Hotel	1316	RBS	0.03	0.20	0.22
		Lite-LM	0.32	0.97	0.70

		Base-LM	0.43	0.97	0.75
		LLM	0.56	0.89	0.87
		Confidence Cascade	0.49	0.86	0.83
		Parallel Ensemble	0.65	0.94	0.92
		Ideal	0.75	0.93	0.93
		Proposed	0.53	0.93	0.87
		RBS	0.25	0.36	0.44
		Lite-LM	0.65	0.98	0.70
Restaurant	1157	Base-LM	0.68	0.98	0.71
		LLM	0.67	0.89	0.83
		Confidence Cascade	0.72	0.85	0.87
		Parallel Ensemble	0.82	0.97	0.93
		Ideal	0.87	0.94	0.93
		Proposed	0.70	0.95	0.87
		RBS	0.03	0.27	0.30
		Lite-LM	0.29	0.96	0.47
Train	1593	Base-LM	0.34	0.96	0.64
		LLM	0.69	0.87	0.87
		Confidence Cascade	0.65	0.85	0.85
		Parallel Ensemble	0.80	0.92	0.89
		Ideal	0.83	0.92	0.90
		Proposed	0.68	0.91	0.85
		RBS	0.25	0.25	-
		Lite-LM	1.00	1.00	-
Taxi	290	Base-LM	1.00	1.00	-
		LLM	0.61	0.61	-
		Confidence Cascade	0.93	0.93	-
		Parallel Ensemble	0.97	0.97	-
		Ideal	1.00	1.00	-
		Proposed	0.85	0.85	-
		RBS	0.11	0.19	0.33
		Lite-LM	0.42	0.98	0.57
Attraction	961	Base-LM	0.65	0.98	0.72
		LLM	0.64	0.96	0.75
		Confidence Cascade	0.58	0.82	0.77
		Parallel Ensemble	0.76	0.99	0.86
		Ideal	0.85	0.98	0.86
		Proposed	0.65	0.98	0.83
		RBS	1.00	1.00	-
		Lite-LM	1.00	1.00	-
No Domain	1121	Base-LM	1.00	1.00	-
		LLM	0.99	0.99	-
		Confidence Cascade	1.00	1.00	-
		Parallel Ensemble	1.00	1.00	-
		Ideal	1.00	1.00	-
		Proposed	1.00	1.00	-
		RBS	1.00	1.00	-
		Lite-LM	1.00	1.00	-

To assess whether differences in Semantic Accuracy were statistically significant, we conducted two-sided exact McNemar tests on utterance-level correctness, with Holm correction applied across the three pairwise comparisons. The proposed method was compared with Confidence Cascade, Parallel Ensemble and Embedding MLP. The corresponding test statistics and adjusted p -values are reported in Table 7. Here, m_{10} denotes

utterances correctly processed by the proposed method but not by the comparison method, and n_{01} denotes the reverse. In the comparison with Confidence Cascade, n_{10} was 733 and n_{01} was 608. Thus, the proposed method correctly processed 125 more utterances than Confidence Cascade processed incorrectly than the reverse, and this difference remained statistically significant after Holm correction ($p = 0.0014$). This result shows that the proposed pre-inference routing procedure, which uses difficulty features to select a model before candidate-model inference, produced correct semantic outputs for more utterances than the confidence-based cascade. In the comparison with Parallel Ensemble, n_{01} was 768 and n_{10} was 149, and the difference was statistically significant ($p < 0.0001$). Parallel Ensemble executes all candidate models for each utterance and determines the final output from their results. Therefore, the larger number of utterances correctly processed only by Parallel Ensemble is consistent with its use of outputs from the entire model pool. However, executing all candidate models resulted in a Total Latency of 4217.85 ms, compared with 806.75 ms for the proposed method. In the comparison with Embedding MLP, n_{01} was 560 and n_{10} was 465. Thus, Embedding MLP correctly processed 95 more utterances than the proposed method processed incorrectly than the reverse, and this difference was also statistically significant after Holm correction ($p = 0.0033$). Embedding MLP uses utterance embeddings as input and learns nonlinear decision boundaries through multiple neural-network layers. This model structure is consistent with its higher Semantic Accuracy than the proposed method. Overall, the McNemar test results provide statistical evidence for the observed differences in Semantic Accuracy between the proposed method and the three comparison methods. Specifically, although the proposed method and Confidence Cascade showed comparable Semantic Accuracy, the proposed method produced more uniquely correct predictions than Confidence Cascade at the utterance level, and the difference was statistically significant according to the McNemar test. This result indicates that the comparable aggregate performance was accompanied by a statistically significant difference in utterance-level correctness in favor of the proposed method on the evaluated data.

Table 7. Pairwise McNemar Test Results for Utterance-Level Semantic Correctness.

Comparison	n_1	n_{10}	p -Value	Adjusted p -Value
Proposed vs. Confidence Cascade	608	733	0.0007	0.0014
Proposed vs. Parallel Ensemble	768	149	<0.0001	<0.0001
Proposed vs. Embedding MLP	560	465	0.0033	0.0033

4.4. Routing Behavior and Efficiency Analysis

This section analyzes the distribution of the final selected models, the structural latency induced by different execution paths, and the routing errors of dynamic routing methods. Table 8 compares the accumulated normalized cost of each routing method under three illustrative routing scenarios—Best, Balanced, and Worst—each defined over 100 utterances. These scenarios are theoretical configurations designed to demonstrate how routing distribution affects accumulated cost. Confidence Cascade, which relies on model confidence for routing decisions, accumulates the computational cost of lower-level models as the final selected model level increases. In the Balanced scenario, the accumulated cost of Confidence Cascade is 61.00, whereas the proposed method incurs a cost of only 40.65, including feature extraction and router overhead, because it executes only the selected model once. In the Worst scenario, where all utterances are routed to the LLM, the cost gap between Confidence Cascade and the proposed method increases to

approximately 1.6 times. In contrast, the cost of the parallel ensemble is accumulated over all candidate models because all models perform inference for each input. Thus, Parallel Ensemble always incurs the same total cost of 161.10, regardless of the scenario, because it requires inference from all models as well as additional ensemble overhead. This result confirms that Parallel Ensemble is the least efficient method in terms of computational cost.

Table 8. Illustrative Scenario-Based Accumulated Normalized Cost per 100 Utterances.

Scenario	Confidence Cascade	Parallel Ensemble	Proposed	Routing Distribution
Best	1.00	161.10	1.40	RBS 100%
Balanced	61.00	161.10	40.65	Each 25%
Worst	161.00	161.10	100.40	LLM 100%

This stepwise accumulation also applies to latency. Figure 3 compares the domain-wise total latency across routing methods with SemAcc reported in the bar labels. To avoid interpreting latency reduction independently of semantic correctness, each bar label reports total latency in milliseconds on the first line and SemAcc on the second line. Ideal denotes the latency obtained when the lowest-capacity model that produces a semantically correct output is selected post hoc for each utterance using ground-truth correctness information. Therefore, Ideal serves as an idealized reference for correctness-preserving model selection, rather than a strict lower bound on latency. Since a routing method may select a lower-capacity model at the expense of semantic correctness, latency should be interpreted jointly with SemAcc and routing error rates. For example, in the Attraction domain, the proposed method recorded lower latency than Ideal; however, this was associated with an 11 percentage-point decrease in SemAcc. This suggests that the proposed method reduced latency by routing more utterances to lower-capacity models, but that this efficiency gain was accompanied by a loss in semantic correctness. Confidence Cascade can achieve low latency when utterances are accepted by lower-level models at early stages. However, as the number of utterances escalated to Base-LM or LLM increases, latency accumulates due to repeated execution of preceding lower-level models. In contrast, the proposed method executes only the selected model once, thereby eliminating redundant lower-level inference in domains that require high-capacity models. As a result, the proposed method reduced total latency by 142.56 ms and 185.45 ms in the complex Hotel and Restaurant domains, respectively. These results suggest that the proposed method can reduce unnecessary inference delay in complex domains, although such latency gains should be interpreted together with SemAcc and routing error patterns.

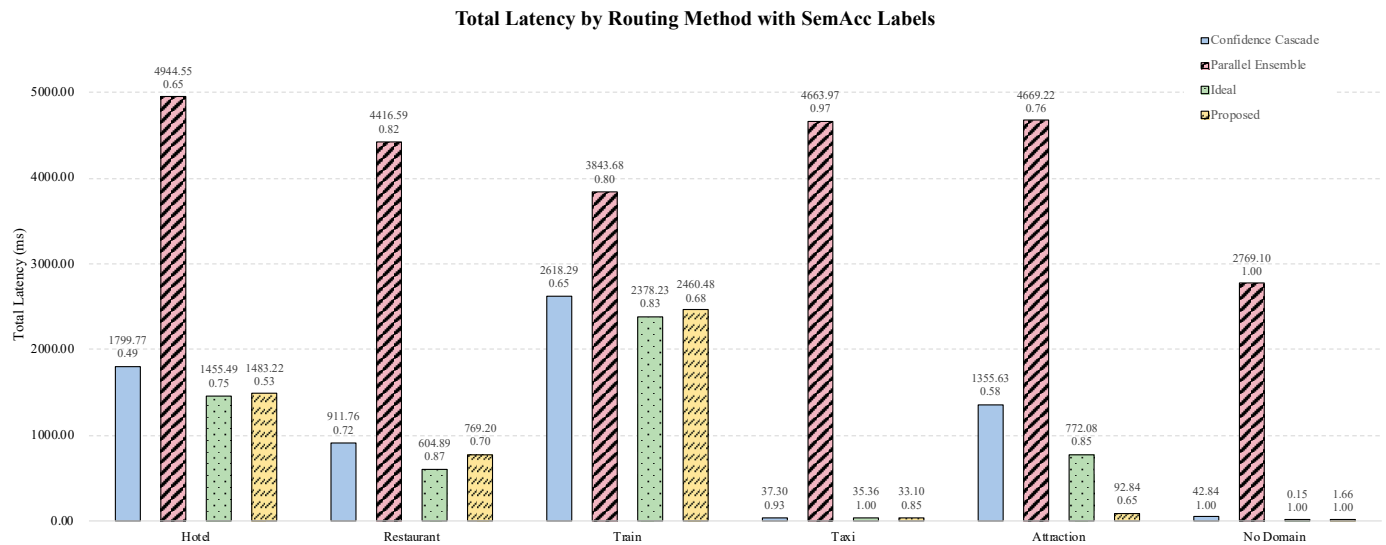


Figure 3. Total Latency (ms) Comparison by Domain and Routing Method.

To provide a more detailed analysis of total latency, Table 9 decomposes the turn-level average latency across all test domains into feature extraction, routing decision, and model inference latency. The proposed method introduces additional feature extraction and routing decision overhead before model inference; however, these overheads remain small compared with the inference latency saved by reducing unnecessary high-capacity model invocation. In contrast, confidence cascading mainly increases model inference latency because multiple models may be executed sequentially before termination. For Parallel Ensemble, since all candidate models are invoked concurrently, the total latency is determined by the inference time of the largest-capacity model, in addition to a routing decision overhead incurred by the MLP-based model selector. For Ideal, model selection is determined post hoc based on ground-truth correctness rather than at inference time; consequently, no feature extraction or routing decision overhead is incurred, and only model inference latency is reported. Overall, the Proposed method achieves the lowest total latency among all methods despite the additional feature extraction and routing decision steps, as it avoids invoking high-capacity models for utterances that do not require them. As discussed in Section 4.3, the lower latency of the proposed method relative to Ideal resulted from routing some utterances to lower-capacity models without access to ground-truth correctness information, and this latency reduction was accompanied by lower Semantic Accuracy.

Table 9. Average Latency Decomposition by Routing Method (ms).

Routing Method	Feature Extraction	Routing Decision	Model Inference	Total Latency
Confidence Cascade	-	<0.01	1127.10	1127.10
Parallel Ensemble	-	0.28	4217.57	4217.85
Ideal	-	-	874.37	874.37
Proposed	0.28	0.93	805.52	806.75

Since latency depends on the level of the model selected by each routing method, we compare the routing distribution of the four methods to analyze how model selection affects overall latency. Figure 4 presents the domain-wise routing distribution for each dynamic routing method. Each stacked bar shows the percentage of utterances assigned to each model level. Since Confidence Cascade performs early acceptance when the output confidence of a model exceeds a predefined threshold, a considerable number of utterances in complex domains such as Hotel and Train tended to be routed to lower-level

models, including RBS and Lite-LM. In contrast, the proposed method recognizes NLU difficulty before inference and therefore avoids invoking RBS in the Hotel, Train, and Attraction domains, instead routing most utterances to Base-LM or LLM. Conversely, in the No Domain category, the proposed method routed 98% of utterances to RBS, thereby reducing unnecessary high-capacity model invocation for simple utterances. In the Attraction domain, the proposed method routed no utterances to the LLM and instead selected Base-LM for most cases, resulting in lower latency. This behavior can be attributed to the fact that the SemAcc gap between Base-LM and the LLM is small in this domain. Rather than invoking the highest-capacity model, the proposed router selected a Base-LM-centered operating point based on pre-inference complexity features. In contrast, Ideal serves as an Ideal upper-bound because it selects the model after observing the actual correctness of each model prediction. Therefore, Ideal can identify utterances for which the LLM provides semantic correctness that lower-level models do not achieve. The proposed method, however, does not rely on such post-hoc correctness information. Instead, it learns generalized patterns of NLU difficulty from input-level features and selects a model level before inference. This difference indicates that the proposed method achieves a favorable empirical trade-off among Semantic Accuracy, latency, and normalized cost under the evaluated pre-inference routing setting.

Table 10 shows that the proposed method reduces under-routing in several complex domains. In Hotel, under-routing decreased from 0.31 to 0.21, and in Attraction, it decreased from 0.27 to 0.19. In Restaurant, under-routing also decreased slightly from 0.16 to 0.14. These results suggest that, although the changes in Semantic Accuracy vary across domains, the proposed method can lower the likelihood of assigning difficult utterances to insufficient lower-capacity models in complex domains. Although this relationship differed in some domains, the tendency was generally consistent with the higher SemAcc of the proposed method reported in Table 10. Overall, Confidence Cascade achieves lower average latency when many utterances are accepted by lower-level models, but this can increase under-routing and reduce SemAcc. The proposed method reduces under-routing in several complex domains by selecting higher-capacity models more conservatively, which improves SemAcc at the expense of increased over-routing. Confidence Cascade can provide an advantage in terms of average latency when a large proportion of utterances are accepted early by lower-level models. However, because this structure relies on post-hoc model confidence, under-routing may occur when the confidence score does not sufficiently reflect the semantic correctness of the prediction, resulting in a degradation of SemAcc. The model selection distribution in Figure 4, where Confidence Cascade routes a relatively large proportion of utterances to lower-level models such as RBS and Lite-LM, indicates the under-routing risk inherent in confidence-based routing. In contrast, the proposed method performs routing decisions at the pre-inference stage using only utterance complexity features extracted from the input utterance and dialogue context. By learning model selection according to utterance complexity in advance, the proposed method reduces under-routing and maintains SemAcc comparable to that of the LLM-only and Confidence Cascade methods. Although the proposed method shows increased over-routing in some domains, this can be interpreted as a trade-off for preventing SemAcc degradation. Parallel Ensemble achieves the best results in terms of SemAcc and routing errors, including both under-routing and over-routing, because it comprehensively explores the inference results of all models. However, due to its requirement to execute the entire model pool, it incurs more than twice the latency of the other methods, making it less efficient in terms of both time and computational cost. Overall, the proposed pre-inference NLU difficulty-aware routing method suppresses under-routing and maintains Semantic Accuracy comparable to the LLM-only and Confidence Cascade methods, while minimizing total latency and cost through single-model routing.

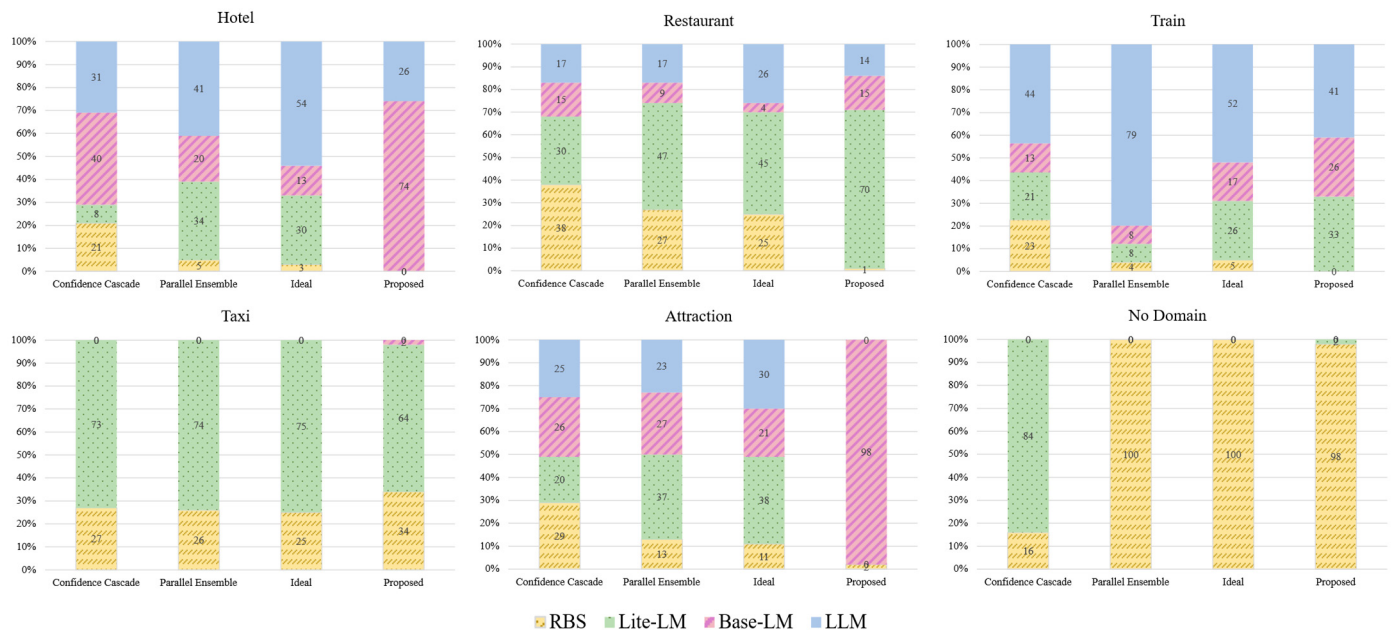


Figure 4. Routing Distribution by Domain and Method.

Table 10. Domain-Wise Routing Performance and Error Analysis.

Domain	Method	SemAcc	Total Latency (ms)	UnderRouting	OverRouting
Hotel	Confidence Cascade	0.49	1799.77	0.31	0.22
	Parallel Ensemble	0.65	4944.55	0.09	0.07
	Proposed	0.53	1483.22	0.21	0.48
Restaurant	Confidence Cascade	0.72	911.76	0.16	0.14
	Parallel Ensemble	0.82	4416.59	0.03	0.03
	Proposed	0.70	769.20	0.14	0.38
Train	Confidence Cascade	0.65	2618.29	0.15	0.09
	Parallel Ensemble	0.80	3843.68	0.03	0.01
	Proposed	0.68	2460.48	0.13	0.18
Taxi	Confidence Cascade	0.93	37.30	0.07	0.05
	Parallel Ensemble	0.97	4663.97	0.03	0.01
	Proposed	0.85	33.10	0.03	0.16
Attraction	Confidence Cascade	0.58	1355.63	0.27	0.34
	Parallel Ensemble	0.76	4669.22	0.06	0.12
	Proposed	0.65	92.84	0.19	0.56
No Domain	Confidence Cascade	1.00	42.84	0.00	0.84
	Parallel Ensemble	1.00	2769.10	0.00	0.00
	Proposed	1.00	1.66	0.02	0.02

In summary, the analyses in Figure 4, Tables 8 and 10 demonstrate that the proposed method alleviates the accumulated computational overhead, which is a structural limitation of post-hoc cascading, while mitigating the under-routing risk caused by confidence-based early acceptance. Although Parallel Ensemble and Ideal achieve high SemAcc by leveraging post-hoc information more comprehensively, they incur substantial cost and latency, thereby exhibiting limitations in terms of cost and latency efficiency. In contrast, the proposed method performs single-model selection using only pre-inference NLU difficulty estimation and achieves a favorable empirical trade-off between Semantic Accuracy and inference efficiency. This confirms that the proposed architecture achieves a favorable trade-off between Semantic Accuracy and Inference Efficiency.

4.5. Comparison with Alternative Pre-Inference Routers

To examine the effects of the input representation and router architecture under the same pre-inference routing setting, this section compares the proposed sequential router with three alternative designs: MultiClass DT, Embedding DT, and Embedding MLP. Table 11 compares the overall routing performance and error patterns of the proposed method with those of the alternative pre-inference routing baselines, using SemAcc, Total Latency, UnderRouting, and OverRouting aggregated across all evaluation domains. Table 11 shows that Embedding MLP achieved the highest SemAcc of 0.75 and the lowest UnderRouting rate of 0.10 among the four methods. However, its Total Latency was 1452.64 ms, approximately 80% higher than that of the proposed method, reflecting the additional computation required for sentence-embedding extraction and neural routing. The proposed method achieved a comparable SemAcc of 0.74 with a Total Latency of 806.75 ms, and also achieved higher SemAcc than both MultiClass DT at 0.64 and Embedding DT at 0.68, while maintaining the lowest latency among the four methods. The proposed method showed a higher OverRouting rate of 0.30 than the other three methods, which ranged from 0.17 to 0.19. Because UnderRouting assigns an utterance to a model below the minimum sufficient level, it has a higher risk of producing an incorrect NLU result. The proposed router therefore reduces UnderRouting while selecting the lowest model level that satisfies the accuracy constraint and limits latency; as a result, some utterances are assigned above the minimum sufficient level and counted as OverRouting, which is interpreted as a consequence of this selection strategy. Unlike Embedding-DT and Embedding-MLP, which route based on dense sentence-embedding representations, the proposed method uses eight explicitly defined difficulty features and sequential DT gates, allowing each routing decision to be traced to specific feature values and gate-level outputs corresponding to task-level, contextual-level, or surface-level signals.

Table 11. Comparison of Routing Performance and Error Analysis with Alternative Pre-inference Routing Baselines.

Method	SemAcc	Total Latency (ms)	UnderRouting	OverRouting
MultiClass DT	0.64	1135.72	0.23	0.19
Embedding DT	0.68	1077.11	0.18	0.18
Embedding MLP	0.75	1452.64	0.10	0.17
Proposed	0.74	806.75	0.12	0.30

4.6. Utterance Complexity Feature Analysis

To analyze which utterance complexity features contribute to routing decisions, we computed feature importance for the Sequential Router in each domain. Since the proposed router consists of multiple binary Decision Tree gates, feature importance was first computed separately for each gate. The gate-level importance scores were then aggregated within each domain to identify the utterance complexity features that consistently contributed to model-level sufficiency prediction. Figure 5 reports the top three features with the highest aggregated importance for each domain. For domains with large slot spaces, such as Hotel, Restaurant, Train, and Attraction, task-level features, including Slot-space Ambiguity and Transaction Risk, exhibited higher importance. This suggests that the Sequential Router primarily exploits complexity signals associated with NLU prediction complexity when determining whether a low-capacity model is sufficient. By contrast, in relatively simple domains such as Taxi and No Domain, surface-level and contextual features, such as length score and topic return, became more influential in addition to Transaction Risk. These results indicate that the model-level requirement in simpler

domains can often be identified using lexical cues rather than complex task-level ambiguity.

To further examine the contribution of the three difficulty classes, we conducted a feature-class removal ablation. The detailed results are reported in Appendix D and Table A4, showing that the removal of each feature class affects semantic performance and routing behavior differently.

Overall, the feature analysis demonstrates that the proposed router makes model-level decisions based on interpretable NLU difficulty signals rather than relying on post-hoc model confidence. Furthermore, the domain-dependent importance of task-level, contextual, and surface-level features indicates that the router uses different difficulty cues depending on the semantic and contextual characteristics of each domain. These findings suggest that input-observable complexity features can serve as a basis for selecting an appropriate NLU model before inference.

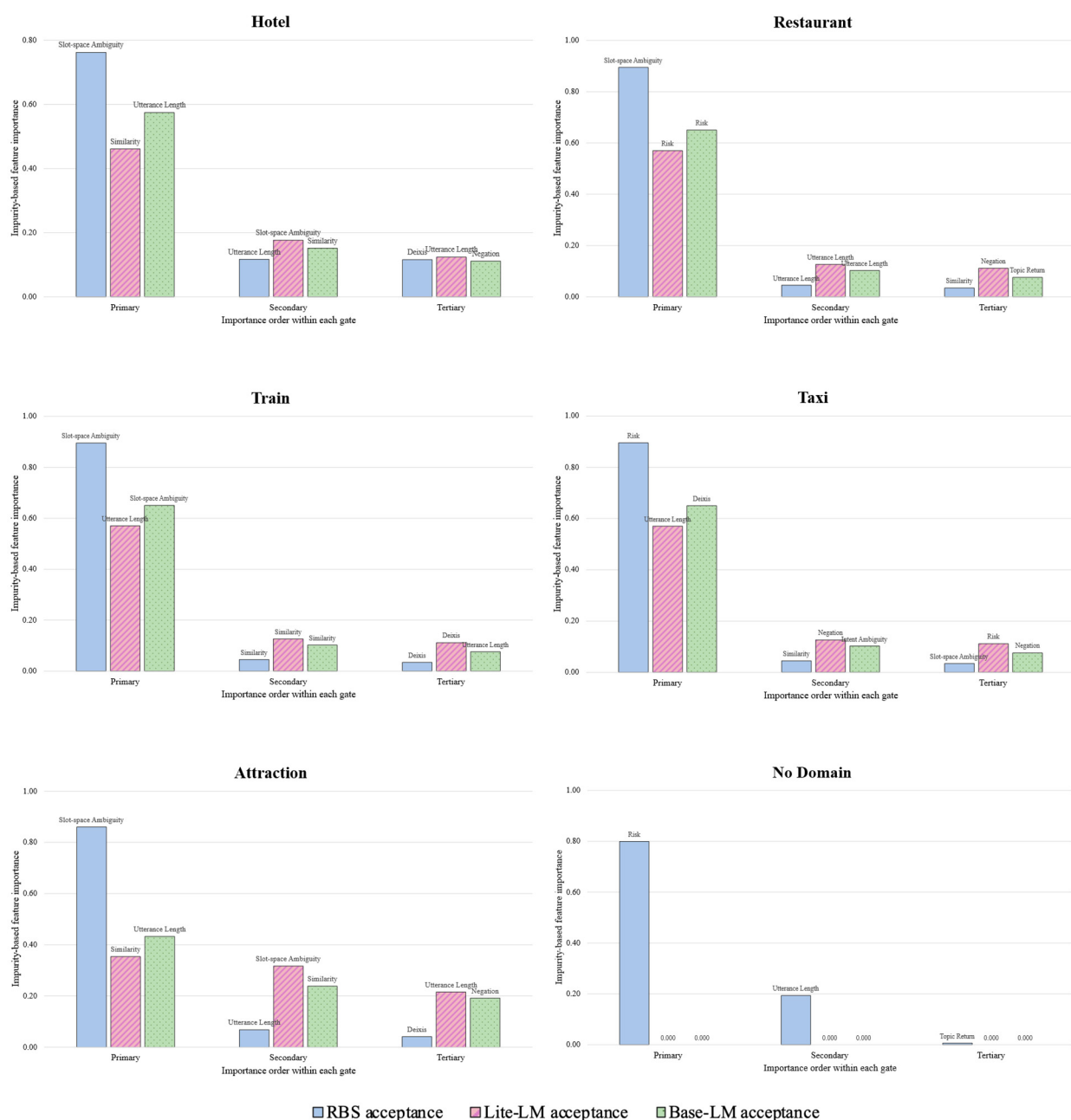


Figure 5. Analysis of Feature Importance for Utterance Complexity.

5. Discussion

The results support the central hypothesis of this study: input-observable NLU difficulty can provide a practical basis for model selection before inference. Unlike post-hoc cascading methods that depend on confidence or uncertainty after model execution, the proposed framework estimates utterance difficulty from the input itself and selects a model level in advance. The comparable Semantic Accuracy and reduced latency indicate that pre-inference routing can avoid redundant computation without relying on sequential trial-and-escalation. In the evaluated setting, the proposed method reduced the average total latency from 1127.10 ms for the confidence cascade to 806.75 ms, a difference of 320.35 ms per utterance, while achieving Semantic Accuracy of 0.74 compared with 0.73 for the cascade. In a real-world conversational system, a similar per-utterance reduction could be expected to shorten the cumulative response time over successive dialogue turns. However, the confidence cascade is not inherently slower under all configurations and may provide different accuracy–latency characteristics depending on its thresholds and execution environment.

The results further indicate that utterance-level NLU difficulty is closely related to the complexity of structured intent-slot prediction. In complex domains, performance losses were associated more with incomplete slot-value prediction than with intent misclassification, suggesting that pre-inference model selection should account for the joint difficulty of intent recognition and slot extraction. The routing error analysis further supports the proposed pre-inference routing strategy, showing that input-observable difficulty features can identify utterances requiring higher-capacity semantic processing before model execution. This is particularly important in task-oriented NLU.

These findings imply that pre-inference difficulty-aware routing is a viable alternative to confidence-based cascading for real-time conversational expert systems in the evaluated single-domain multi-turn NLU setting. However, the current framework has several limitations. First, the difficulty representation relies on predefined schema information, manually designed lexical rules, and regular expressions. Consequently, adapting the framework to unseen or highly complex domains requires additional domain-specific rule construction and maintenance, which may limit scalability as the number of domains and schema elements increases. Second, because the proposed method is trained offline, it may be difficult to adapt to dynamic changes in user behavior or domain distributions. Third, the current temporal features may not sufficiently represent long-range contextual dependencies in very long multi-turn conversations. Fourth, the current evaluation is based on oracle domain labels and single-domain turns, and therefore does not account for domain-prediction errors or utterances containing intents and slot constraints from multiple domains. Fifth, the current evaluation does not quantify seed-level variability across the trainable model and routing components. Sixth, the reported latency results are specific to the evaluated CPU-based deployment setting, and routing efficiency may vary depending on hardware, batching, parallel execution, and edge AI acceleration.

Future work will extend the proposed framework to predicted-domain and multi-domain settings through a two-stage architecture in which a lightweight domain predictor first identifies one or more candidate domains, after which domain-conditioned intent and slot difficulty features are aggregated for model-level routing. In addition, we will explore automatic schema-guided construction of lexical rules, aliases, and normalization mappings, together with learned difficulty representations, to reduce the manual effort required when adapting the framework to new or complex domains. Further research will formulate utility-based routing by jointly estimating the expected accuracy, cost, and operational overhead of each candidate model. We will also investigate deployment-feedback-based threshold adaptation and transferable difficulty representations for cross-domain and out-of-domain inputs. The latency–accuracy characteristics, conversational

turn-level accumulation, and deployment feasibility of the routing methods will also be compared and optimized in GPU and edge AI accelerator environments under different batching, parallel-execution, and model-optimization settings.

6. Conclusions

In this study, we proposed a pre-inference NLU difficulty-aware routing framework that estimates utterance complexity before executing any candidate NLU model and selects an appropriate model level for multi-turn NLU in single-domain dialogue settings. To mitigate the additional inference cost and accumulated latency introduced by confidence-based cascading methods, we designed a routing method that performs routing prior to candidate-model execution and independently of candidate-model confidence or uncertainty signals. The proposed method performs domain-specific routing decisions using utterance complexity features observable from the input utterance and dialogue context. Experimental results demonstrate that the proposed method reduced high-capacity model invocation and average latency while achieving competitive Semantic Accuracy compared with the few-shot prompted LLM-only baseline and the post-hoc confidence cascade. Comparisons with multiclass and embedding-based pre-inference routers further showed that the proposed method maintained comparable semantic performance while requiring lower end-to-end latency than the evaluated alternatives.

The results of this study suggest that selecting the model level in advance according to utterance-level NLU difficulty can be an efficient alternative to processing all utterances with the same high-performance model in task-oriented conversational systems. Furthermore, by showing that input-level utterance complexity features provide meaningful signals for model selection, this study presents a pre-inference routing direction that differs from conventional output confidence-based cascading. Future work will extend the stability and generalizability of the proposed framework by evaluating predicted-domain and multi-domain settings and by developing learned difficulty representations, adaptive routing-threshold policies, and cross-domain transfer learning for out-of-domain inputs.

Author Contributions: S.Y., D.K. and S.L. conceptualized and designed the experiments; S.Y. designed and implemented the system; S.Y., D.K. and S.L. wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by a 2026 research grant from Sangmyung University (2025-A000-0323).

Data Availability Statement: The dataset is available at <https://github.com/budzianowski/multiwoz> (24 June 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

The lexical and rule-based cues used for feature extraction were constructed from the MultiWOZ 2.2 domain ontology and common surface expressions observed in task-oriented dialogue. These rules were used only to compute pre-inference difficulty features. They were not used as gold labels, and no gold intent or slot-value annotations from the test instances were used during feature computation.

Appendix A.1. Slot Cue Matching

For each domain, the slot cue lexicon was constructed from ontology-defined slot names, ontology values, and common paraphrases. For example, area-related cues include north, south, east, west, and centre; price-related cues include cheap, moderate,

expensive, budget, and price; and booking-related cues include day, time, people, stay, and book.

Appendix A.2. Intent Cue Matching

Intent cues were defined using task-oriented action keywords such as find, search, look for, need, book, reserve, cancel, change, modify, and update.

Appendix A.3. Transaction Risk

Transaction-risk cues include booking, reservation, cancellation, change, modification, confirmation, and related surface forms.

Appendix A.4. Deixis (History Reference) Cue Matching

Deixis cues capture referential expressions that depend on prior dialogue context for resolution, independent of domain ontology. These include pronouns and demonstratives such as it, that, this, these, those, there, and they.

Appendix A.5. Negation Cue Matching

Negation cues capture negated or restrictive expressions that alter the interpretation of user intent, independent of domain ontology. These include negation markers such as no, not, never, without, except, and restrictive quantifiers such as few, hardly, rarely, scarcely, and barely, as well as contracted forms such as n't.

The complete feature-extraction rule sets, aliases, normalization mappings, and regular-expression patterns are available from the corresponding author upon reasonable request.

Appendix B

Table A1 presents the complete prompt template used for LLM-based NLU inference. The bracketed fields were dynamically populated for each test instance using the current user utterance, dialogue history, previous dialogue state, and domain-specific schema information. The same prompt template and output-parsing procedure were applied consistently across all test instances.

Table A1. Prompt Used for LLM-Based NLU Inference.

Prompt
You are an NLU system for task-oriented dialogue. Extract the intent and the slots expressed or changed in the CURRENT user utterance. ## Intents find_hotel, book_hotel, find_restaurant, book_restaurant, find_train, book_train, find_taxi, find_attraction, find_hospital, find_police, find_bus, no_intent ## Valid Slots (ONLY use slots from this list) hotel: hotel-pricerange, hotel-type, hotel-parking, hotel-bookday, hotel-bookpeople, hotel-bookstay, hotel-area, hotel-stars, hotel-internet, hotel-name restaurant: restaurant-food, restaurant-pricerange, restaurant-area, restaurant-name, restaurant-bookday, restaurant-bookpeople, restaurant-booktime

train:

train-departure, train-destination, train-day,
train-arriveby, train-leaveat, train-bookpeople

taxi:

taxi-leaveat, taxi-destination, taxi-departure, taxi-arriveby

attraction:

attraction-type, attraction-area, attraction-name

Rules

- Output ONLY the two lines below. No notes, no explanations, no extra text.
- Use ONLY intents from the Intents list above.
- Use ONLY slot names from the Valid Slots list above.
- Extract the intent from the CURRENT user utterance.
- For slots, return ONLY slots newly mentioned or changed in the CURRENT user utterance.
- Do NOT repeat unchanged slots from [PREV].
- If the user changes a slot value (e.g., “actually make it expensive”), output the slot with the new value.
- If the user explicitly says they do not care about a constraint (e.g., “any area is fine”, “doesn’t matter”, “I don’t care”), output that slot with value “dontcare”.
- If the user mentions a specific entity name from the system’s recommendation, extract it as the name slot (e.g., hotel-name, restaurant-name).
- For unknown values never write “not specified” or “unknown” — omit the slot entirely.
- Use lowercase canonical values: “centre”, “cheap”, “yes”, “no”, “dontcare”, and lowercase day names (“monday”, “tuesday”, etc.).

Input Format

[PREV]: <previous accumulated slots>

SYSTEM: <system utterance>

USER: <current utterance to analyze>

Output Format

intent: <intent_name>

slots: <slot1>=<value1>, <slot2>=<value2>

(If no new or changed slots → slots: none)

Examples

Example 1

[PREV]:

USER: I need a cheap hotel in the centre with free wifi

intent: find_hotel

slots: hotel-pricerange=cheap, hotel-area=centre, hotel-internet=yes

Example 2

[PREV]: hotel-pricerange=cheap, hotel-area=centre, hotel-internet=yes

SYSTEM: I recommend the Lovell Lodge. It has free wifi and is in the centre.

USER: That sounds great. Please book it for 3 people, 2 nights starting Friday.

intent: book_hotel
 slots: hotel-name=lovell lodge, hotel-bookpeople=3, hotel-bookstay=2, hotel-bookday=fri-day

Example 3

[PREV]: hotel-area=centre, hotel-pricerange=cheap

USER: I don't care about the area anymore

intent: find_hotel
 slots: hotel-area=dontcare

Example 4

[PREV]:

USER: I want a restaurant that serves italian food in the centre

intent: find_restaurant
 slots: restaurant-food=italian, restaurant-area=centre

Example 5

[PREV]: restaurant-food=italian, restaurant-area=centre

SYSTEM: I recommend Pizza Hut City Centre.

USER: Book a table for 4 at 19:00 on saturday

intent: book_restaurant
 slots: restaurant-name=pizza hut city centre, restaurant-bookpeople=4, restaurant-book-time=19:00, restaurant-bookday=saturday

Example 6

[PREV]:

USER: I need a train from cambridge to london on friday

intent: find_train
 slots: train-departure=cambridge, train-destination=london, train-day=friday

Example 7

[PREV]:

USER: I also need a taxi from the hotel to the restaurant, leaving by 14:00

intent: find_taxi
 slots: taxi-departure=the hotel, taxi-destination=the restaurant, taxi-leaveat=14:00

Example 8

[PREV]:

USER: Thank you, that's all I need

intent: no_intent
 slots: none

Appendix C

Table A2 reports the final domain- and gate-specific thresholds used by the proposed sequential router. The thresholds represent the minimum predicted sufficiency

probability required to select the corresponding model level at each sequential gate. Candidate thresholds were searched on the development set from 0.05 to 0.95 in increments of 0.05. The selected values were fixed before test-set evaluation. Each threshold was applied to the probability output of the corresponding domain- and gate-specific sufficiency classifier. Because the classifiers were trained separately for each domain and gate, their probability distributions are not necessarily calibrated to a common scale. In addition, Gate 2 and Gate 3 operate only on samples rejected by the preceding gates. Therefore, the numerical threshold values should not be interpreted as directly comparable measures of domain difficulty or model capability across domains and gates. The selected thresholds show that the acceptance criterion varies according to both the domain and the model level. For example, Hotel and Attraction use relatively high thresholds at both the RBS and Lite-LM gates, whereas Taxi and No Domain use lower thresholds across multiple gates. Restaurant uses the highest RBS threshold, while Train uses a relatively high threshold at the Base-LM gate. These values indicate that a single global threshold was not applied uniformly; instead, the operating point of each gate was determined using its development-set routing behavior. Threshold values at the boundaries of the search range, including 0.05 and 0.95, were selected from the predefined candidate grid based on the development-set selection procedure. A threshold of 0.05 means that a sample is assigned to the corresponding model only when it reaches that gate and its predicted sufficiency probability is at least 0.05. A threshold of 0.95 applies the same decision rule with a minimum predicted sufficiency probability of 0.95. These values define gate-specific decision criteria and do not directly represent domain difficulty.

Table A2. Domain-Wise Gate-Specific Thresholds for the Proposed Sequential Router.

Domain	Gate 1 Threshold (RBS)	Gate 2 Threshold (Lite-LM)	Gate 3 Threshold (Base-LM)
Hotel	0.90	0.85	0.40
Restaurant	0.95	0.45	0.30
Train	0.85	0.55	0.50
Taxi	0.45	0.05	0.05
Attraction	0.90	0.85	0.05
No Domain	0.05	0.05	0.05

Table A3 reports the final domain-specific confidence thresholds used by the confidence cascade. Since the RBS does not produce a continuous confidence score, inputs were retained at the RBS stage only when the predefined rules were matched; otherwise, they were forwarded to Lite-LM. Therefore, the RBS threshold is denoted by “–”. At the Lite-LM and Base-LM stages, inputs with post-inference confidence scores below the reported thresholds were forwarded to the next model level. The thresholds in Tables A2 and A3 are applied to different decision signals. In the proposed router, each threshold is applied before candidate-model execution to the estimated sufficiency score produced by the corresponding binary gate. In the confidence cascade, each threshold is applied after Lite-LM or Base-LM inference to the confidence score of the resulting prediction.

Table A3. Domain-Wise Gate-Specific Thresholds for the Confidence Cascade.

Domain	Gate 1 Threshold (RBS)	Gate 2 Threshold (Lite-LM)	Gate 3 Threshold (Base-LM)
Hotel	–	0.90	0.70
Restaurant	–	0.80	0.70
Train	–	0.75	0.70
Taxi	–	0.60	0.80
Attraction	–	0.75	0.70
No Domain	–	0.80	0.85

Appendix D

Appendix D reports a focused feature-group ablation analysis. The analysis examines how removing each difficulty dimension affects semantic performance and routing behavior. We compare the full representation with variants excluding task-level, contextual, or surface-level features.

To examine the contribution of the proposed multi-dimensional difficulty representation, we conducted a leave-one-group-out ablation using the full feature set as the reference. Task-level, contextual, and surface-level feature groups were removed separately, and the sequential router was retrained under the same experimental protocol. This design measures the marginal contribution of each feature group to the full representation, i.e., the extent to which the overall difficulty representation depends on that group. We evaluate how the removal of each group affects overall Semantic Accuracy (SemAcc), Intent Accuracy (IntentAcc), Slot F1, LLM call rate, UnderRouting, and OverRouting.

Table A4 compares the overall semantic performance and routing behavior of the full feature set and the three leave-one-group-out configurations. Removing the task-level features produced the largest performance decrease, with SemAcc declining by 9%p, IntentAcc by 6%p, and Slot F1 by 5%p. This result indicates that task-level features are the most relevant to joint intent–slot difficulty among the three feature groups. Removing the contextual features reduced SemAcc by 5%p, whereas IntentAcc and Slot F1 declined by less than 1%p. Removing the surface-level features reduced all three accuracy metrics by only 1%p, reflecting a smaller contribution to semantic performance. All ablated configurations reduced the LLM call rate by 2–3%p; however, this reduction was accompanied by a 5–9%p increase in UnderRouting. This suggests that the lower LLM call rate did not reflect more accurate routing decisions, but rather a higher rate of assigning difficult utterances to models with insufficient capacity. OverRouting increased by 2%p after removing either the task-level or contextual features, whereas it decreased by 5%p after removing the surface-level features, indicating that surface-level features are more closely associated with model escalation than with semantic accuracy.

Table A4. Overall Results of the Feature-Group Removal Ablation.

Feature Set	SemAcc	IntentAcc	Slot F1	LLM Call Rate	UnderRouting	OverRouting
Full	0.74	0.96	0.85	0.18	0.12	0.30
w/o Task	0.65	0.90	0.80	0.16	0.21	0.32
w/o Contextual	0.69	0.95	0.85	0.15	0.17	0.32
w/o Surface	0.73	0.95	0.84	0.15	0.17	0.25

References

1. Larson, S.; Leach, K. A Survey of Intent Classification and Slot-Filling Datasets for Task-Oriented Dialog. *arXiv* **2022**, <https://doi.org/10.48550/arXiv.2207.13211>.
2. Louvan, S.; Magnini, B. Recent Neural Methods on Slot Filling and Intent Classification for Task-Oriented Dialogue Systems: A Survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online), 8–13 December 2020; Scott, D., Bel, N., Zong, C., Eds.; International Committee on Computational Linguistics: Barcelona, Spain, 2020; pp. 480–496.
3. Li, Y.; Kim, S.-E.; Park, S.-B.; Han, C. MIDAS: Multi-Level Intent, Domain, And Slot Knowledge Distillation for Multi-Turn NLU. In *Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 2025*; Chiruzzo, L., Ritter, A., Wang, L., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 8004–8027.
4. Yang, S.; Kim, D.; Lee, S. HYLR-FO: Hybrid Approach Using Language Models and Rule-Based Systems for On-Device Food Ordering. *Electronics* **2025**, *14*, 775. <https://doi.org/10.3390/electronics14040775>.

5. Ding, D.; Mallick, A.; Wang, C.; Sim, R.; Mukherjee, S.; Rühle, V.; Lakshmanan, L.; Awadallah, A.H. Hybrid Llm: Cost-Efficient and Quality-Aware Query Routing. In *Proceedings of the International Conference on Learning Representations (ICLR 2024)*, 2024.
6. Kossmann, F.; Wu, Z.; Turk, A.; Tatbul, N.; Cao, L.; Madden, S. CascadeServe: Unlocking Model Cascades for Inference Serving. *arXiv* **2024**, <https://doi.org/10.48550/arXiv.2406.14424>.
7. Chun, C.; Rim, D.; Park, J. LLM ContextBridge: A Hybrid Approach for Intent and Dialogue Understanding in IVSR. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, Abu Dhabi, United Arab Emirates, January 2025*; Rambow, O., Wanner, L., Apidianaki, M., Al-Khalifa, H., Eugenio, B.D., Schockaert, S., Darwish, K., Agarwal, A., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 794–806.
8. Chen, L.; Zaharia, M.; Zou, J. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *arXiv* **2023**, <https://doi.org/10.48550/arXiv.2305.05176>.
9. Sharma, A. Improving Intent Classification in an E-Commerce Voice Assistant by Using Inter-Utterance Context. In *Proceedings of the 3rd Workshop on e-Commerce and NLP, Seattle, WA, USA, July 2020*; Malmasi, S., Kallumadi, S., Ueffing, N., Rokhlenko, O., Agichtein, E., Guy, I., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 40–45.
10. Castillo-López, G.; de Chalendar, G.; Semmar, N. Intent Recognition and Out-of-Scope Detection Using LLMs in Multi-Party Conversations. In *Proceedings of the 26th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Avignon, France, August 2025*; Béchet, F., Lefèvre, F., Asher, N., Kim, S., Merlin, T., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 504–512.
11. Arora, G.; Jain, S.; Merugu, S. Intent Detection in the Age of LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, Miami, Florida, USA, January 2024*; Dernoncourt, F., Preotiuc-Pietro, D., Shimorina, A., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 1559–1570.
12. Pattnayak, P.; Agarwal, A.; Meghwani, H.; Patel, H.L.; Panda, S. Hybrid AI for Responsive Multi-Turn Online Conversations with Novel Dynamic Routing and Feedback Adaptation. In *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing, Albuquerque, New Mexico, USA, May 2025*; Shi, W., Yu, W., Asai, A., Jiang, M., Durrett, G., Hajishirzi, H., Zettlemoyer, L., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 215–229.
13. Gupta, N.; Narasimhan, H.; Jitkrittum, W.; Rawat, A.S.; Menon, A.K.; Kumar, S. Language Model Cascades: Token-Level Uncertainty and Beyond. In *Proceedings of the International Conference on Learning Representations 2024 (ICLR 2024)*, 2024; pp. 4147–4180.
14. Hu, Z.; Chen, N.F.; Lee, R.K.-W. Are Current Task-Oriented Dialogue Systems Able to Satisfy Impolite Users? *IEEE Trans. Comput. Soc. Syst.* **2025**, *12*, 2876–2887. <https://doi.org/10.1109/TCSS.2024.3521020>.
15. Atuhurra, J.; Kamigaito, H.; Watanabe, T.; Nichols, E. Domain Adaptation in Intent Classification Systems: A Review. *arXiv* **2024**, <https://doi.org/10.48550/arXiv.2404.14415>.
16. Wu, T.; Wang, M.; Xi, Y.; Zhao, Z. Intent Recognition Model Based on Sequential Information and Sentence Features. *Neurocomputing* **2024**, *566*, 127054. <https://doi.org/10.1016/j.neucom.2023.127054>.
17. Chen, Q.; Zhuo, Z.; Wang, W. BERT for Joint Intent Classification and Slot Filling. *arXiv* **2019**, <https://doi.org/10.48550/arXiv.1902.10909>.
18. Li, D.; Niu, K.; Chen, X.; Wang, Z. A Hybrid Model Based on Graph Convolutional Network and Multi-Head Transformer for Joint Multiple Intent Detection and Slot Filling. In *2024 IEEE International Conference on Big Data (BigData)*; IEEE: New York, NY, USA, 2024; pp. 1049–1056. <https://doi.org/10.1109/BigData62323.2024.10825739>.
19. Qin, L.; Liu, T.; Che, W.; Kang, B.; Zhao, S.; Liu, T. A Co-Interactive Transformer for Joint Slot Filling and Intent Detection. *arXiv* **2021**, <https://doi.org/10.48550/arXiv.2010.03880>.
20. Han, S.C.; Long, S.; Li, H.; Weld, H.; Poon, J. Bi-Directional Joint Neural Networks for Intent Classification and Slot Filling. *arXiv* **2022**, <https://doi.org/10.48550/arXiv.2202.13079>.
21. Hossain, M.M.; Blanco, E. Leveraging Affirmative Interpretations from Negation Improves Natural Language Understanding. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*; Goldberg, Y., Kozareva, Z., Zhang, Y., Eds.; Association for Computational Linguistics: Abu Dhabi, United Arab Emirates, 2022; pp. 5833–5847.
22. Chen, Y.; Luo, Z. Pre-Trained Joint Model for Intent Classification and Slot Filling with Semantic Feature Fusion. *Sensors* **2023**, *23*, 2848. <https://doi.org/10.3390/s23052848>.
23. Ong, I.; Almahairi, A.; Wu, V.; Chiang, W.-L.; Wu, T.; Gonzalez, J.E.; Kadous, M.W.; Stoica, I. RouteLLM: Learning to Route LLMs with Preference Data. *arXiv* **2025**, <https://doi.org/10.48550/arXiv.2406.18665>.
24. Zhong, V.; Xiong, C.; Socher, R. Global-Locally Self-Attentive Dialogue State Tracker. *arXiv* **2018**, <https://doi.org/10.48550/arXiv.1805.09655>.

25. Rana, M.; Hacıoglu, K.; Gopalan, S.; Boothalingam, M. Zero-Shot Slot Filling in the Age of LLMs for Dialogue Systems. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, Abu Dhabi, United Arab Emirates, January 2025*; Rambow, O., Wanner, L., Apidianaki, M., Al-Khalifa, H., Eugenio, B.D., Schockaert, S., Darwish, K., Agarwal, A., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 697–706.
26. Henderson, M.; Thomson, B.; Williams, J.D. The Second Dialog State Tracking Challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), Philadelphia, PA, USA, June 2014*; Georgila, K., Stone, M., Hastie, H., Nenkova, A., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2014; pp. 263–272.
27. Dhole, K.D. Resolving Intent Ambiguities by Retrieving Discriminative Clarifying Questions. *arXiv* **2024**, <https://doi.org/10.48550/arXiv.2008.07559>.
28. Bihani, G.; Rayz, J.T. Fuzzy Classification of Multi-Intent Utterances. In *Proceedings of the North American Fuzzy Information Processing Society Annual Conference*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 37–51.
29. Song, M.; Yu, B.; Li, Q.; Wang, Y.; Liu, T.; Xu, H. Enhancing Joint Multiple Intent Detection and Slot Filling with Global Intent-Slot Co-Occurrence. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, February 2022*; Goldberg, Y., Kozareva, Z., Zhang, Y., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2022; pp. 7967–7977.
30. Zang, X.; Rastogi, A.; Sunkara, S.; Gupta, R.; Zhang, J.; Chen, J. MultiWOZ 2.2: A Dialogue Dataset with Additional Annotation Corrections and State Tracking Baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, Online, July 2020*; Wen, T.-H., Celikyilmaz, A., Yu, Z., Papangelis, A., Eric, M., Kumar, A., Casanueva, I., Shah, R., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 109–117.
31. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv* **2020**, <https://doi.org/10.48550/arXiv.1910.01108>.
32. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, <https://doi.org/10.48550/arXiv.1907.11692>.
33. Firdaus, M.; Bhatnagar, S.; Ekbal, A.; Bhattacharyya, P. A Deep Learning Based Multi-Task Ensemble Model for Intent Detection and Slot Filling in Spoken Language Understanding. In *Proceedings of the Neural Information Processing*; Cheng, L., Leung, A.C.S., Ozawa, S., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 647–658.
34. Firdaus, M.; Bhatnagar, S.; Ekbal, A.; Bhattacharyya, P. *Intent Detection for Spoken Language Understanding Using a Deep Ensemble Model*; Geng, X., Kang, B.-H., Eds.; Springer International Publishing: Cham, Switzerland, 2018; Volume 11012, pp. 629–642.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.