Coordinating Multi-Assistant Interaction in Smart Homes: A Front-End Delegation Approach

Pedro Carneiro
IEETA, DETI, University of Aveiro, Portugal
LASI – Intelligent System Associate Laboratory, Guimarães, Portugal

Inês Águia
IEETA, DETI, University of Aveiro, Portugal
LASI – Intelligent System Associate Laboratory, Guimarães, Portugal

Nuno Almeida
IEETA, DETI, University of Aveiro, Portugal
LASI – Intelligent System Associate Laboratory, Guimarães, Portugal

António Teixeira
IEETA, DETI, University of Aveiro, Portugal
LASI – Intelligent System Associate Laboratory, Guimarães, Portugal

Abstract

Smart home environments increasingly integrate multiple domain-specific assistants, yet interaction remains fragmented across heterogeneous interfaces. Users are required to adapt to different systems and interaction modalities, which introduces complexity and reduces accessibility, particularly in scenarios such as active ageing.

This work aims to design a unified conversational abstraction that enables seamless interaction with multiple specialised assistants through a single entry point. The goal is to decouple user interaction from underlying services while preserving modularity and extensibility.

We present the `Butler Assistant`, a modular conversational framework that orchestrates multiple domain-specific assistants within a unified architecture. The system introduces a routing layer based on a shared registry and supports interchangeable routing engines. Two approaches are explored: a lexical model based on Naive Bayes and a semantic model based on sentence embeddings. The system is evaluated under both in-domain and out-of-domain scenarios.

The evaluation shows that the semantic model improves routing accuracy in domains with limited training data, demonstrating stronger generalization capabilities. In contrast, the lexical model is more sensitive to training data distribution, achieving strong performance in well-represented domains but degrading in low-resource scenarios. Importantly, both approaches exhibit reliable behaviour in out-of-domain conditions, consistently redirecting unsupported inputs to fallback outcomes and avoiding incorrect domain assignments. These results highlight the complementary strengths of lexical and semantic routing and emphasize the role of system-level decision mechanisms in ensuring safe and robust conversational interaction.

2012 ACM Subject Classification Human-centered computing → Natural language interfaces; Computing methodologies → Natural language processing; Human-centered computing → Interaction paradigms

Keywords and phrases Conversational Assistants, Smart Homes, Naïve Bayes, Embedding-driven Semantic Similarity

Digital Object Identifier 10.4230/OASIcs.SLATE.2026.16

Funding This research was supported by projects Casa Viva+ and VITALITY (COMPETE2030-FEDER-00755100 16385) and Research Unit IEETA, funded by National Funds through the FCT – Foundation for Science and Technology, in the context of the project UIDB/00127/2020.

1 Introduction

Conversational assistants now serve as primary interfaces for interacting with smart home environments, providing natural, hands-free access to a wide range of domestic functions. These systems support a range of everyday activities, including control of domotic systems, kitchen assistance, access to contextual information such as weather forecasts, and guidance during physical exercise. This breadth of functionality positions conversational assistants as key enablers of intuitive and pervasive interaction [17, 3]. This relevance is amplified in households with older adults or individuals receiving home health services, where these interfaces can reduce cognitive load, promote autonomy, and facilitate aging in place [18, 13].

Smart homes oriented toward health monitoring increasingly integrate sensing infrastructures, activity recognition, and AI-based feedback mechanisms to support well-being and chronic disease management [19, 9]. When integrated with conversational interaction, these systems can provide accessible and personalized support in healthcare, especially for self-management, monitoring, and patient education [4, 10].

However, current commercial assistants remain confined to proprietary ecosystems such as Alexa, Google Assistant, and Siri. Each is optimized for a narrow set of capabilities and lacks interoperability across heterogeneous devices and services [7, 8].

This parallelism of smart homes, where multiple assistants coexist (e.g., competing wake words leading to misrecognition), exacerbates fragmentation in complex setups that require coordinated control [8]. A challenge manifests acutely in real-world living lab environments. In these innovative scenarios, which simulate daily life with health monitoring, users encounter difficulties with unified dialogue management. This limitation inhibits the effectiveness of aging-in-place support.

To address these limitations, this paper proposes a Modular Conversational Interaction Framework for Smart Homes, an architecture that aggregates multiple conversational assistants into a cohesive and extensible system. Implemented and evaluated in a living lab context, the framework enables coordinated domotics control, multimodal kitchen assistance, and weather information services. The primary objective is to demonstrate how modular aggregation overcomes interoperability barriers, enhances user experience, and establishes a robust foundation for smart homes that support daily living and health needs, particularly for older adults in home-based care.

2 Related Work

Multi-Domain Conversational Assistants – Research on conversational agents has progressively moved from single-purpose systems toward multi-domain assistants that can manage heterogeneous tasks and services through a unified dialogue interface. A key milestone in this direction is the Schema-Guided Dialogue dataset, which was introduced to support scalable multi-domain conversational agents and zero-shot generalization to unseen services [20]. Building on this, research has continued to explore dialogue modeling strategies that improve robustness in multi-domain settings, including modular architectures and more flexible dialogue state tracking. Collectively, these works demonstrate that scalable conversational interaction requires not only language understanding, but also structured handling of domain-specific knowledge and uncertainty [21, 12].

Multi-Agent and Multi-Assistant Orchestration – A complementary direction is the study of how multiple conversational assistants can be combined into a single user-facing system. Recent work on multi-assistant conversational AI indicates that users benefit when orchestration is managed through a unified interface, rather than requiring them to

choose between assistants [6]. This highlights the value of a coordination layer that can route, aggregate, and mediate among specialized assistants. The challenge extends beyond response generation to include determining which assistant should handle a request, when clarification is needed, and how to manage fallback behavior across different components.

Conversational Interfaces in Smart Homes and Homecare – Building on the coordination challenges outlined above, conversational interfaces have also become increasingly relevant in smart home and home care environments, where users interact with devices, services, and monitoring functions through voice or text. Recent work has shown that voice assistants and conversational user interfaces can improve accessibility and natural interaction in domestic settings, especially when users need to control multiple services without dealing with complex interfaces [1, 16]. However, much of this work still relies on a single assistant tightly coupled to a specific ecosystem or vendor platform, which limits interoperability and extensibility. For that reason, smart-home conversational interaction remains an active research area, particularly in scenarios where multiple services must coexist within a single interaction layer [2, 11].

Conversational Agents for Health Monitoring and Support – Similarly, in health-related settings, conversational agents have been explored for monitoring, education, support, and patient engagement. A literature review of conversational agents in health care reported growing interest in these systems, while also highlighting fragmentation across use cases, dialogue strategies, and evaluation methods [4]. More recent studies continue to emphasize both the potential and the challenges of integrating conversational agents into health workflows, including issues of trust, usability, and professional and patient adoption [14, 15]. Our own work has explored conversational assistants integrated with smart mirrors for mental health assessment [22, 5], alongside the development of a kitchen assistant aimed at supporting active ageing at home, promoting independence through assistance with everyday tasks such as meal preparation and resource management.

Overall, the literature provides strong foundations in multi-domain dialogue modeling, multi-agent coordination, smart-home conversational interaction, and health-supportive conversational agents. Nevertheless, most existing systems remain specialized: they either focus on one assistant, one domain, or one service ecosystem. To address this gap, this work proposes an orchestration layer that integrates multiple specialized conversational assistants, providing a unified interface for smart environments while maintaining back-end modularity. By combining routing, dispatch, and execution mechanisms, users can interact with different assistants without needing to know which service or domain is responsible for each request.

3 Proposed Solution – A Delegation Approach

The proposed framework adopts a delegation-based architecture in which a central conversational agent coordinates user requests and delegates them to specialized domain assistants. This design reflects the Butler like role of the central Assistant, acting as the main entry point of the system, responsible for interpreting high-level intents, maintaining dialogue flow, and forwarding requests to the appropriate component.

This framework is inspired by the organizational structure of traditional manor houses, a model often associated with the upstairs-downstairs hierarchy. In this metaphor, the `butler` acts as the primary interface between the user and the system, overseeing all interactions and managing the distribution of tasks across the available assistants. As Mr. Carson states: *«I am the butler. This is the house I run. Upstairs and down, it is my responsibility»*[1].

[1] *Downton Abbey Script Book Season 1* (HarperCollins, 2012), p. 45.

Supporting the butler are `servant` assistants, each specialized in a specific smart home domain such as domotics control, kitchen assistance, and weather information services, within a Portuguese-language conversational interaction environment.. These assistants operate through a common routing structure that enables systematic task delegation and consistent response handling.

A key advantage of this approach is modularity: new assistants can be introduced or existing ones improved independently, without affecting the central conversational pipeline. The routing layer supports this extensibility by providing structured decisions, confidence scores, and ranked candidate assistants, which allows the decision process to select the most appropriate handler for each request.

Overall, this delegation approach explores coordination, specialization, and separation of concerns. By separating user-facing interaction from domain-specific processing, the framework contributes to a unified conversational experience while preserving flexibility and scalability for future smart home extensions.

4 Architecture and Processing Pipeline

This section presents the Butler Assistant as a pipeline-oriented architecture consisting of 4 processing stages (`Input Reception` → `Routing` → `Decision Process` → `Dispatch`).

4.1 Stage 1: Input Reception

The first stage defines the entry point of the Butler Assistant, where user utterances are received. Inputs are assumed to be preprocessed by upstream modalities (e.g., Automatic Speech Recognition) and are therefore provided as textual sequences. Each utterance is assigned a unique `request_id`, enabling end-to-end traceability throughout the pipeline. This identifier allows tracking the processing flow across components and collecting metrics for analysis and evaluation.

4.2 Stage 2: Routing

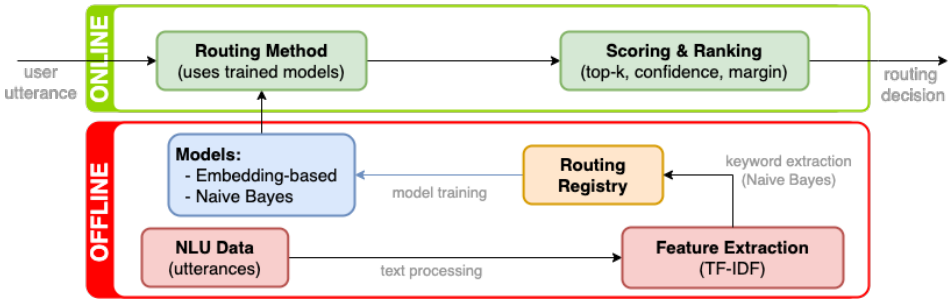


Figure 1 Overview of the routing process, distinguishing between the offline and online phases.

The routing stage is the central linguistic component of the Butler Assistant. Its goal is to determine which domain-specific assistant is the most appropriate handler for a given user utterance by comparing the input against a structured registry of examples, thresholds, and rule-based signals. Rather than returning a simple label, the routing layer produces a structured result containing the selected route, a confidence score, and a ranked list of candidate assistants.

As shown in Figure 1, the routing process is organized into two distinct phases:

- **Offline Phase** – This phase is executed prior to deployment and is responsible for constructing the Routing Registry and training the routing models.
- **Online Phase** – At runtime, incoming user utterances are processed by a routing engine, selected at system initialization, which generates a routing decision based on selected engine.

4.3 Stage 3: Decision Process

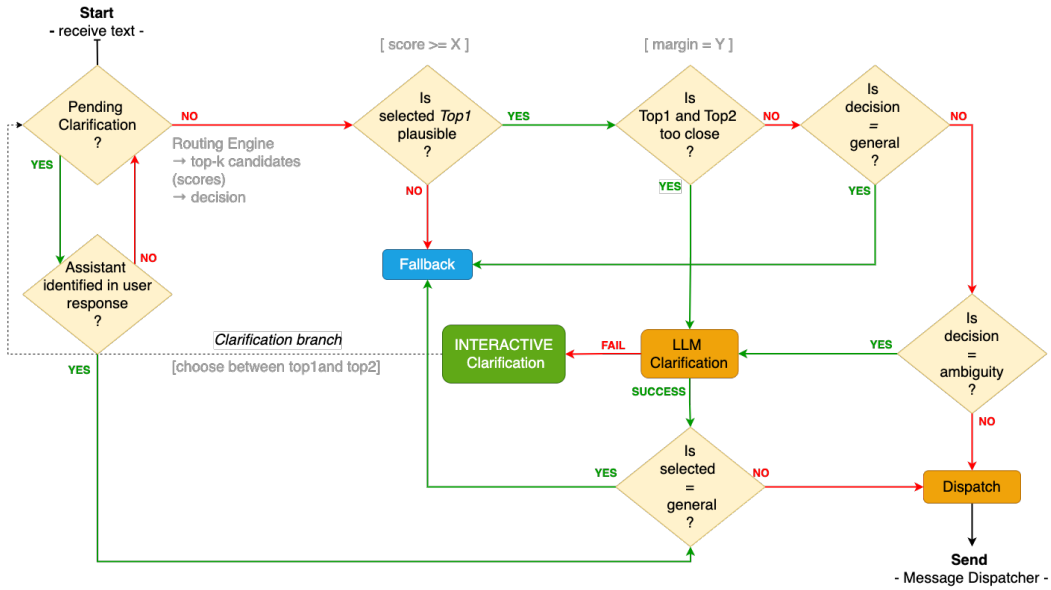


Figure 2 Decision flow implemented. For this proposal, $X = 0.25$ represents the minimum plausibility score required for delegation, while $Y = 0.08$ corresponds to the ambiguity margin between candidate assistants..

The `Decision Process`, illustrated in Figure 2, defines how routing outputs are transformed into system actions. Rather than performing linguistic classification, this module evaluates routing confidence and applies system-level control rules to determine the next interaction step. This next step corresponds to a small set of observable outcomes that define the system behaviour at runtime, as summarized in Table 1.

Table 1 Decision Process decision outcomes and their interpretation.

Outcome	Description
`dispatch`	Routed to a domain-specific assistant
`fallback`	Handled by the general assistant due to (low confidence/no match)
`clarify`	System requests disambiguation due to competing candidates
`no_prediction`	No decision could be produced due to system or processing failure

These outcomes provide a direct mapping between routing signals and system behaviour, and are later used in the evaluation phase to analyse both routing correctness and overall pipeline performance.

4.4 Stage 4: Dispatch

After the `Decision Process` determines the final action, the request is forwarded to the `Message Dispatcher`. This component delivers the utterance to the selected domain-specific assistant and returns the response through the same communication path.

5 Implementation

This section details the implementation of the routing architecture, describing how knowledge is structured, how routing decisions are computed, and how the system resolves uncertainty during interaction. The goal is to provide a clear view of the components that enable efficient, reliable, and context-aware assistant selection.

5.1 Routing Registry

The `Routing Registry` is the central knowledge base of the routing system. It is automatically generated from RASA Natural Language Unit (NLU) files and contains the available routes (the assistants), their representative examples, confidence thresholds, and rule-based signals. The registry also defines global configuration values (Table 2) shared across methods, such as the default threshold, ambiguity margin, top-k value, and fallback route. Because the registry acts as the source of knowledge for the routing process, it defines the available routes, examples, and configuration parameters that guide routing behaviour, while the models determine how routing decisions are computed based on this information.

Table 2 Configurable parameters controlling routing behavior.

Parameter	Value	Description
Global threshold	0.6	Minimum confidence required for a valid match
Ambiguity margin	0.08	Minimum score difference between top candidates
Top-K candidates	3	Number of candidate routes returned by the router
Boosting weight	0.1	Weight applied to rule-based keyword boosting

In addition to example-based representations, the registry incorporates keyword-based rules generated through a statistical feature selection approach. By transforming training utterances into a Term Frequency-Inverse Document Frequency TF-IDF) representation and applying a Multinomial Naive Bayes model, the system identifies highly discriminative terms for each route. These rules complement the underlying routing models by reinforcing strong lexical signals.

This design is driven by the constraints of real-world conversational systems, where routing must be both efficient and responsive to maintain a fluid interaction. In addition, domain-specific assistants typically provide only a limited number of training examples, making it impractical to rely solely on data-driven approaches. As a result, lightweight models are complemented with rule-based signals, enabling the system to incorporate discriminative cues without increasing training complexity.

5.2 Routing Methods

The `Routing Method` implement a common interface and transforms user input into a standardized routing decision, expressed through a small set of possible outcomes (Table 3). This abstraction allows the system to support different linguistic approaches while preserving

the same output format and downstream behavior. In the current architecture, the Butler Assistant supports two engines: a probabilistic Naive Bayes model and a semantic similarity model based on sentence embeddings.

Table 3 Routing decision outcomes produced by the routing method.

Decision	Description
`match`	A route exceeds the confidence threshold and is selected
`ambiguous`	Multiple routes are plausible with insufficient margin
`no_match`	No route meets the minimum confidence threshold

In addition to model-based scoring, both routing methods incorporate a rule-based boosting mechanism, derived from the routing registry. This mechanism reinforces highly discriminative lexical signals, complementing the statistical or semantic representation used by each model.

The training process follows the routing registry configuration, which is intentionally imbalanced: `domotic` (144 examples), `kitchen` (55), and `weather` (15).

5.2.1 Naive Bayes

The Naive Bayes (NB) Model formulates the routing task as a probabilistic text classification problem, where a user utterance x is assigned to a route $r \in \mathcal{R}$ by maximizing the posterior probability:

$$r^* = \arg\max_{r \in \mathcal{R}} P(r \mid x)$$

Under the Multinomial Naive Bayes assumption, this posterior is computed as:

$$P(r \mid x) \propto P(r) \prod_{i=1}^{n} P(w_i \mid r)$$

where w_i are the tokens extracted from the utterance x. Utterances are represented using a TF-IDF encoding, which captures term importance across routes based on the registry.

At inference time, the model computes a probability distribution over routes, interpreted as base scores. To enhance discrimination, a rule-based boosting mechanism $b(r \mid x)$ is applied. This component assigns additional weight to routes whose associated keywords (derived from the registry) are present in the input utterance, reinforcing strong lexical cues that may not be fully captured by the probabilistic model alone. The influence of this mechanism is controlled by a global parameter (Table 2).

$$s(r \mid x) = s_{\text{base}}(r \mid x) + b(r \mid x)$$

Candidate routes are ranked, and decisions (Table 3) are made based on the global threshold θ and the ambiguity margin δ.

5.2.2 Embedding-based

The Embedding-based Model implements a semantic similarity problem, representing both user utterances and training examples as dense vectors in a shared embedding space.

Let $\phi(x) \in \mathbb{R}^d$ denote the embedding of an utterance x, computed using a pre-trained sentence encoder. During the offline phase, all example utterances associated with each route r are encoded to form a set of embeddings $\{\phi(x_j^r)\}$.

At inference time, an utterance x is encoded, and its similarity to each route is computed using cosine similarity:

$$\mathrm{sim}(x, x_j^r) = \frac{\phi(x) \cdot \phi(x_j^r)}{\|\phi(x)\| \, \|\phi(x_j^r)\|}$$

The base score for a route is defined as the maximum similarity across its examples:

$$s_{\mathrm{sem}}(r \mid x) = \max_j \ \mathrm{sim}(x, x_j^r)$$

This nearest-neighbour strategy captures semantic proximity, enabling the system to generalize beyond surface-level lexical variations and correctly handle paraphrases.

As in the lexical model, a rule-based boosting mechanism $b(r \mid x)$ is applied to reinforce discriminative lexical cues derived from the routing registry. This yields a final score:

$$s(r \mid x) = s_{\mathrm{sem}}(r \mid x) + b(r \mid x)$$

Candidate routes are ranked according to their final scores, and the same decisions (Table 3) are made based on the global threshold θ and the ambiguity margin δ.

5.3 Decision Process

The decision process can be interpreted as a set of conditional steps, summarized as follows:

Pending Clarification – If the system is in a clarification state, the incoming user input is interpreted as a response to a previous disambiguation request. In this case, the decision process directly resolves the decision based on the clarified intent, bypassing standard routing evaluation.

Confidence Check – If no clarification is pending, the decision process evaluates whether the top-ranked candidate meets the minimum confidence threshold (`X` in Figure 2). If no route satisfies this condition, the request is redirected to the fallback mechanism, typically handled by the general assistant.

Decision Interpretation – Following routing, the decision process interprets the predicted decision type. If the decision corresponds to a fallback-related outcome (e.g., `general`), the request is redirected accordingly. If the decision indicates ambiguity, the clarification strategy is triggered.

Ambiguity Detection – When the top candidate exceeds the threshold but the score difference between the top candidates is below the ambiguity margin (`Y`), the input is considered ambiguous. This distinction separates ambiguity detection at the routing level from decision-making at the system level.

Clarification Strategy – In ambiguous cases, the decision process attempts automatic clarification using contextual information. If ambiguity persists, the system enters an interactive clarification state, prompting the user to disambiguate the request. The next user input is then treated as a clarification response.

Final Decision – Once a decision is resolved, the decision process produces a structured `next_step` object that determines how the system proceeds (Listing 1). If the resolved decision corresponds to the general route, fallback handling is applied; otherwise, the request is dispatched to the selected assistant.

Listing 1 Example of a `next_step` object, for *utterance* = Can you turn off all the house lights?

```
"next_step" : {
        "action":"dispatch",
        "assistant":"domotic",
        "message":{
                "type":"dispatch", "assistant":"domotic",
                "text":"Podes desligar todas as luzes da casa?"},
        "meta":{
                "reason":"top_candidate_accepted",
                "confidence":0.8534,
                "top_k":[
                        0:{ "route_id":"domotic",
                             "score":0.8534,
                             "base_score":0.7534,
                             "boost_score":0.1},
                        1: (...)
                        3:{ "route_id":"weather",
                             "score":0.0239,
                             "base_score":0.0239,
                             "boost_score":0} ]},
        "request_id":"req_1e687e73dfe4"}
```

6 Evaluation Process

Two experimental scenarios were considered to assess routing performance under different settings:

User input limited to the topics covered by servant assistants (`In-domain`): Considers utterances drawn exclusively from the domains covered by the system servant assistants (kitchen, domotic, and weather).

Inclusion of user input not covered by servant assistants (`Out-of-domain`): This setting extends the first by considering also utterances that do not belong to any of the domains supported by servant assistants. These examples simulate unsupported or unexpected user inputs.

The test sets used in both scenarios were constructed from NLU examples that were not used during model training, ensuring a clear separation between training and evaluation data. For the out-of-domain scenario, additional utterances were obtained from the NLU data of an external assistant covering a different domain.

All utterances were processed through the full Butler Assistant pipeline **with Interactive clarification disabled**, ensuring consistent batch evaluation. Each request includes an `expected_route`, enabling supervised evaluation. An observability layer was used to record the outputs of each pipeline stage and capture detailed timing information. This enables fine-grained analysis of system behaviour, including routing decisions, pipeline execution, and end-to-end latency.

7 Results

This section details the experimental results across two distinct scenarios. We begin by evaluating routing performance under both in-domain and out-of-domain conditions, followed by a comparative analysis of system latency.

7.1 Routing Performance: In-domain

The performance (accuracy) of the two routing methods in the `In-Domain` scenario, evaluated at both the routing stage and the final pipeline stage is presented in Figure 3.

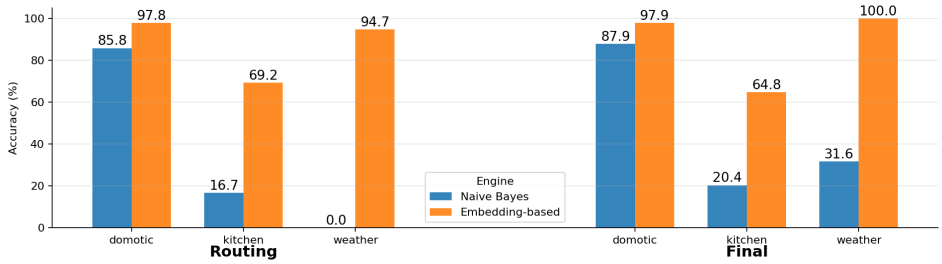
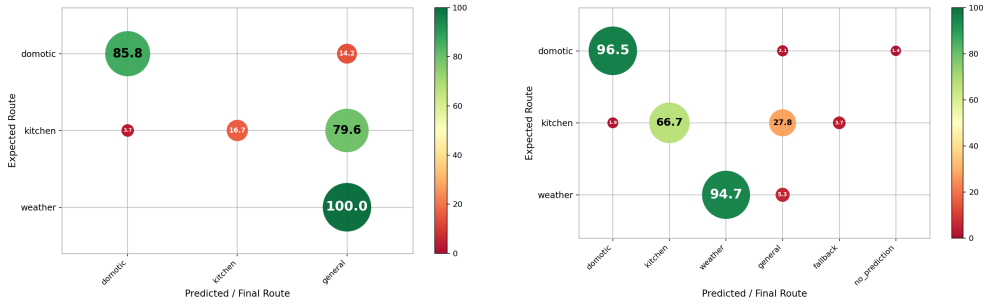


Figure 3 Accuracy per expected route for both routing output and final pipeline output in the `In-Domain` scenario.

A clear and consistent difference between the two routing methods can be observed and the embedding-based model significantly outperforms the Naive Bayes model across all domains. For the embedding-based model, accuracy remains consistently high in both stages, with values close to 100% in the `domotic` and `weather` domains, and above 65% in `kitchen`. The differences between routing and final accuracy are minimal, indicating that correct routing decisions are preserved throughout the pipeline and that downstream components introduce negligible degradation. In contrast, the Naive Bayes model exhibits substantial variability across domains. While performance is relatively strong in `domotic` (approximately 86–88%), it degrades significantly in `kitchen` (around 17–20%) and `weather` (0–32 %). This behaviour highlights a strong dependence on training data distribution, with performance decreasing sharply in low-resource domains. Additionally, the gap between routing and final accuracy is limited for the Naive Bayes model, suggesting that most errors originate at the routing stage and are not corrected by subsequent components in the pipeline.

To complement the accuracy metric, confusion matrices were also calculated and analyzed (Figure 4). To ensure the analysis remains independent of dataset imbalance, values are normalized as percentages per expected route, enabling a direct comparison across domains with different sample sizes.



(a) Naive Bayes.

(b) Embedding-based.

Figure 4 Graphical representation of the confusion matrices for the two routing methods in the in-domain scenario. Values are row-normalized.

The routing behaviour varies significantly across domains, particularly for the Naive Bayes model, while remaining relatively consistent for the embedding-based approach.

For the Naive Bayes model, performance is highly dependent on the target domain. The best results are observed in the `domotic` domain, where most utterances are correctly routed (approximately 85%), reflecting the higher number of training examples. In contrast, performance degrades considerably in `kitchen`, where only a small fraction of utterances is correctly classified, and a large proportion is incorrectly mapped to the `general` route. This effect becomes even more pronounced in the `weather` domain, where the model fails to assign any utterance to the correct route, effectively treating all inputs as `general`. This indicates that, under low-resource conditions, the Naive Bayes model is unable to capture sufficient lexical evidence and defaults to fallback behaviour. In contrast, the embedding-based model exhibits consistently strong performance across all domains. High accuracy is achieved not only in `domotic`, but also in `kitchen` and `weather`, despite the significant differences in training data availability. Errors are relatively limited and more evenly distributed, with only a small proportion of utterances mapped to the `general` route. This suggests that semantic representations allow the model to generalize beyond surface-level lexical patterns, maintaining reliable routing behaviour even in low-resource domains.

7.2 Routing Performance: Out-of-Domain

The performance (accuracy) for the two routing methods on the `Out-Domain` scenario at the various stages is presented in Figure 5.

In this setting, accuracy is computed by considering utterances that do not belong to any supported domain. Each request is associated with an expected out-of-domain label, and correct behaviour corresponds to safely redirecting the input to fallback-related outcomes (`general`, `fallback`, `no_prediction`), rather than assigning it to a domain-specific assistant.

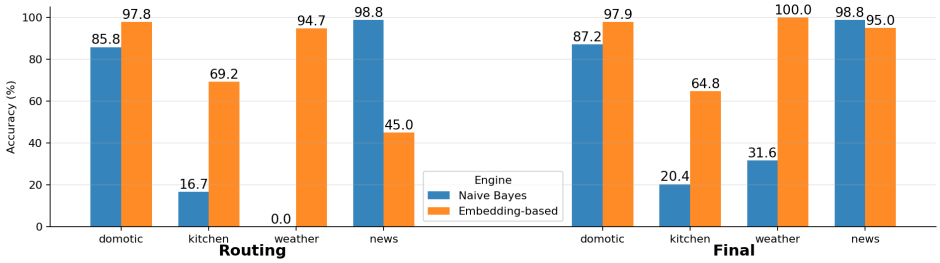


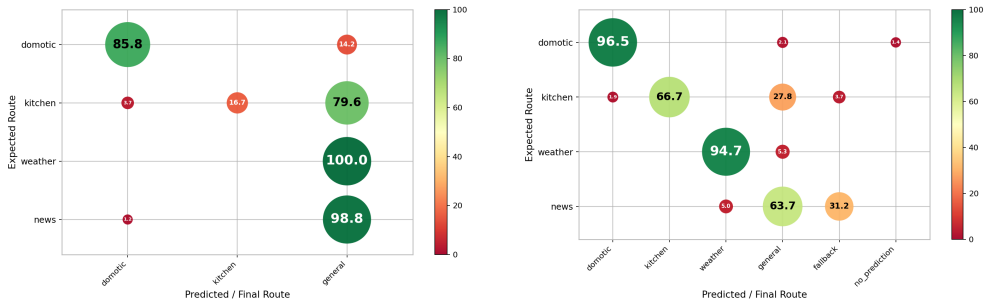
Figure 5 Accuracy per expected route for both routing output and final pipeline output in the `Out-Domain` scenario with unsupported domain (`news`).

Compared with the `In-Domain` setting, this scenario reveals a more differentiated behaviour across routes and across stages of the pipeline. In particular, the main question is not only whether the methods preserve good performance in supported domains, but also whether unsupported inputs (`news`) are correctly handled by the decision process.

For the Naive Bayes model, performance remains highly uneven across routes. The best result is obtained for `news`, where accuracy is already near-perfect at the routing stage and remains so at the final stage. In contrast, the weakest result is observed for `weather`, where routing accuracy drops to zero and only partially recovers in the final output. The `kitchen` route also remains low, while `domotic` preserves comparatively strong performance. Overall, this confirms that the Naive Bayes method remains strongly dependent on route-specific

lexical evidence, producing highly variable results across assistants. For the embedding-based model, routing accuracy remains strong for the supported domains, with very high values in `domotic` and `weather`, and clearly better performance than Naive Bayes in `kitchen`. However, the most interesting behaviour is observed for `news`: while routing accuracy is substantially lower at the routing stage, it increases sharply at the final stage. This shows that, in the `Out-Domain` setting, the decision process plays an important corrective role for the embedding-based method, helping transform uncertain routing outputs into safer final decisions. This contrasts with the `In-Domain` scenario, where the difference between routing and final accuracy was generally small for both methods. Here, the downstream decision logic has a more visible impact, particularly for the embedding-based approach in the unsupported `news` case.

To complement the accuracy analysis, row-normalized confusion matrices were obtained for the out-of-domain scenario, as shown in Figure 6. These matrices provide a detailed view of how unsupported inputs are distributed across predicted routes.



(a) Naive Bayes.

(b) Embedding-based.

Figure 6 Graphical representation of the confusion matrices for the two routing methods in the out-of-domain scenario. Values are row-normalized.

The analysis focuses on the `news` category, which represents the out-of-domain inputs. In this context, correct behaviour corresponds to mapping these utterances to fallback-related outcomes (`general`, `fallback`, `no_prediction`), rather than dispatching them to a domain-specific assistant.

For the Naive Bayes model, the behaviour is highly conservative. Unsupported `news` utterances are almost entirely mapped to `general`, with negligible assignment to other domains. This confirms that the model rarely attempts semantic generalization and instead defaults to safe rejection when lexical evidence is insufficient. As a result, it avoids incorrect routing to domain-specific assistants, but at the cost of limited flexibility. In contrast, the embedding-based model exhibits a more distributed behaviour. While the majority of `news` utterances are correctly mapped to `general`, a small proportion is assigned to semantically related domains, such as `weather`. This reflects the model's ability to capture semantic similarity, but also highlights a tendency toward over-generalization when handling unsupported inputs.

More importantly, these observations align with the stage-level accuracy results. Although the embedding-based model may initially produce ambiguous or incorrect routing decisions for out-of-domain inputs, the decision process mitigates these effects, increasing the proportion of correct final outcomes. Conversely, the Naive Bayes model shows minimal change between routing and final stages, indicating that its behaviour is largely determined at the routing level. This indicates that the system is able to correctly reject unsupported inputs, avoiding incorrect dispatch to domain-specific assistants.

7.3 Latency

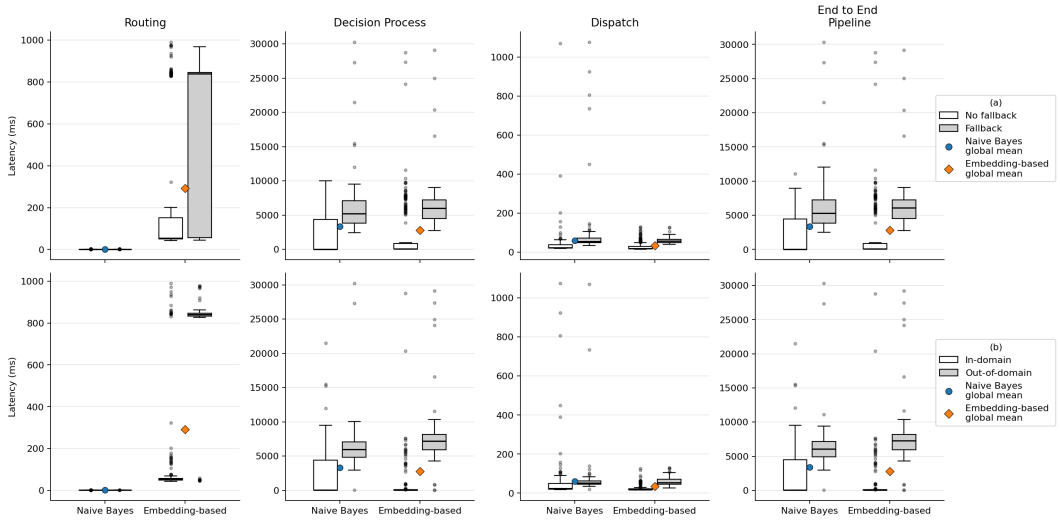


Figure 7 Latency for the pipeline stages. Top row presents results as function of routing method (x-axis) and activation or not of fallback during decision process. Bottom row presents results as function of the evaluation scenario (`In-Domain` vs `Out-of-Domain`). Boxplots are complemented with markers to ensure legibility of the visualizations, different y-axis ranges were applied to the individual stages..

An examination of both mean and median latencies shows that the system's stages generally operate within time frames compatible with a usable conversational assistant, responding quickly enough to sustain natural dialog between users and the assistant. Across all configurations, median latency remains low for most components, particularly in the `Routing` and `Dispatch` stages, confirming that the core pipeline consistently meets the responsiveness requirements of real-time interaction.

The latency analysis further reveals a clear distinction between typical system behaviour and occasional high-cost interactions. While most interactions are processed quickly, a small number of cases exhibit significantly higher latency. When comparing interactions that involve fallback with those that do not (shown in the top row of Figure 7) it becomes evident that high-latency cases occur exclusively under the `Fallback` condition. When examining the bottom row of the plot, which separates in-domain and out-of-domain scenarios, similar high-latency cases appear in both conditions. The same pattern appears under both routing approaches (`Naive Bayes` and `Embedding-based`), confirming that extreme latency values arise from the fallback mechanism rather than from the choice of routing strategy.

The results make clear that the `Decision Process` stage is the primary contributor to end-to-end latency. Whereas routing and dispatch remain reliably fast, the decision component introduces significant fluctuations, most notably in fallback situations. These fluctuations propagate downstream, causing the overall latency to depend chiefly on the cost of decision-making under uncertainty. The presence of these high-latency cases can therefore be directly attributed to the fallback mechanism, which relies on external LLM-based processing. Unlike the rest of the pipeline, this component is less controllable because it depends on external conditions and resources such as model inference time, network latency, and variability in upstream services. These dependencies explain why fallback interactions can lead to occasional but substantially slower responses compared with typical system behaviour.

These results show that responsiveness is limited by uncertain inputs, not routing. Reducing fallbacks and streamlining downstream decisions (especially those using external services) is key to lowering latency.

8 Conclusion and Future Work

The challenge of providing a unified conversational interface over domain-specific assistants was addressed by ensuring that routing decisions remain accurate, robust, and safe under both in-domain and out-of-domain conditions. To tackle this problem, a modular routing architecture was proposed in which user interaction is decoupled from underlying services through a structured decision layer, where model-based routing is combined with explicit system-level policies for ambiguity handling and fallback control.

Superior performance was achieved by the embedding-based approach, especially in low-resource domains where its ability to generalize beyond lexical similarity proved advantageous. Furthermore, both routing methods exhibited stable behaviour in out-of-domain conditions, consistently avoiding mis-dispatches through fallback activation.

In terms of latency, the system maintains response times appropriate for real-time use, with end-to-end latency shaped mainly by pipeline-level operations rather than by the routing methods.

Future Work

While the results are encouraging, a number of limitations still remain. Naïve Bayes is sensitive to vocabulary coverage, while semantic models may introduce ambiguity through similarity-based matching. These results stress the importance of explicit uncertainty-handling mechanisms, such as ambiguity detection and fallback, to maintain reliable behaviour in multi-domain systems. Aligned with these results, a number of promising avenues for further development become apparent:

- **Improvement of Fallback** – Exploring faster and less network dependent alternatives.
- **Hybrid Fusion Model** – Develop learned fusion strategies that dynamically combine lexical and semantic information, enabling the system to balance robustness and generalization depending on the input characteristics.
- **Adaptive decision** – Extend the decision layer to incorporate adaptive mechanisms that adjust routing thresholds and ambiguity margins based on observed system behaviour. This includes dynamically tuning confidence thresholds and decision boundaries according to performance metrics, enabling the system to better handle variability across domains and improve robustness over time.
- **Decomposition of Complex Commands** – Extend the system to handle more complex user utterances by incorporating techniques such as chunking and/or syntactic/semantic parsing. This will enable the system to decompose a single input into multiple actionable commands and route each one to the appropriate servant assistant.

References

1 Saul Albert, Magnus Hamann, and Elizabeth Stokoe. Conversational user interfaces in smart homecare interactions: A conversation analytic case study. In *Proceedings of the 5th International Conference on Conversational User Interfaces*, CUI '23, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3571884.3597140.

2 Anastasios Alexiadis, Alexandros Nizamis, Ioannis Koskinas, Dimosthenis Ioannidis, Konstantinos Votis, and Dimitrios Tzovaras. Applying an Intelligent Personal Agent on a Smart Home Using a Novel Dialogue Generator. In Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, pages 384–395, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-49186-4_32.

3 Frank Bentley, Chris Luvogt, Max Silverman, Rushani Wirasinghe, Brooke White, and Danielle Lottridge. Understanding the Long-Term Use of Smart Speaker Assistants. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(3):91:1–91:24, 2018. doi:10.1145/3264901.

4 Lorainne Tudor Car, Dhakshenya Ardhithy Dhinagaran, Bhone Myint Kyaw, Tobias Kowatsch, Shafiq Joty, Yin-Leng Theng, and Rifat Atun. Conversational Agents in Health Care: Scoping Review and Conceptual Analysis. *Journal of Medical Internet Research*, 22(8):e17158, 2020. doi:10.2196/17158.

5 Pedro Carneiro, Inês Águia, David Palricas, Ana Patrícia Rocha, António Teixeira, and Nuno Almeida. Kitchen Assistant for Active Ageing at Home. In *Proceedings of the 11th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*, DSAI '24, pages 193–202, New York, NY, USA, 2025. Association for Computing Machinery. doi:10.1145/3696593.3696624.

6 Christopher Clarke, Joseph Peper, Karthik Krishnamurthy, Walter Talamonti, Kevin Leach, Walter Lasecki, Yiping Kang, Lingjia Tang, and Jason Mars. One Agent To Rule Them All: Towards Multi-agent Conversational AI. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3258–3267, Dublin, Ireland, 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.findings-acl.257.

7 European Commission. Final report on consumer Internet of Things sector inquiry. URL: https://ec.europa.eu/commission/presscorner/detail/en/ip_22_402.

8 Colin Crawford. Protocol power: Matter, IoT interoperability, and a critique of industry self-regulation, 2024. doi:10.14763/2024.2.1776.

9 Inge Dhamanti, Ika M. Nia, Krishnaraj Nagappan, and Balaji P. Srikaanth. Smart home healthcare for chronic disease management: A scoping review. *DIGITAL HEALTH*, 9:20552076231218144, 2023. doi:10.1177/20552076231218144.

10 Anh Duong and Maria Valero. Usability of Voice Assistants in Healthcare: A Systematic Literature Review. In Dario Salvi, Pieter Van Gorp, and Syed Ahmar Shah, editors, *Pervasive Computing Technologies for Healthcare*, pages 386–401, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-59717-6_25.

11 Maksym Ketsmur, António Teixeira, Nuno Almeida, and Samuel Silva. Towards European Portuguese Conversational Assistants for Smart Homes. In Ricardo Rodrigues, Jan Janoušek, Luís Ferreira, Luísa Coheur, Fernando Batista, and Hugo Gonçalo Oliveira, editors, *8th Symposium on Languages, Applications and Technologies (SLATE 2019)*, volume 74 of *Open Access Series in Informatics (OASIcs)*, pages 5:1–5:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.SLATE.2019.5.

12 Lizi Liao, Tongyao Zhu, Le Hong Long, and Tat Seng Chua. Multi-domain Dialogue State Tracking with Recursive Inference. In *Proceedings of the Web Conference 2021*, WWW '21, pages 2568–2577, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3442381.3450134.

13 Mingzhou Liu, Caixia Wang, and Jing Hu. Older adults; intention to use voice assistants: Usability and emotional needs. *Heliyon*, 9, November 2023. doi: 10.1016/j.heliyon.2023.e21932. doi:10.1016/j.heliyon.2023.e21932.

14 A. Luke MacNeill, Lillian MacNeill, Alison Luke, and Shelley Doucet. Health Professionals' Views on the Use of Conversational Agents for Health Care: Qualitative Descriptive Study. *Journal of Medical Internet Research*, 26(1):e49387, 2024. doi:10.2196/49387.

15 Laura Martinengo, Xiaowen Lin, Ahmad Ishqi Jabir, Tobias Kowatsch, Rifat Atun, Josip Car, and Lorainne Tudor Car. Conversational Agents in Health Care: Expert Interviews to Inform the Definition, Classification, and Conceptual Framework. *Journal of Medical Internet Research*, 25(1):e50767, 2023. doi:10.2196/50767.

16 Bettina Minder, Patricia Wolf, Matthias Baldauf, and Surabhi Verma. Voice assistants in private households: a conceptual framework for future research in an interdisciplinary field. *Humanities and Social Sciences Communications*, 10(1):173, 2023. doi:10.1057/s41599-023-01615-z.

17 Martin Porcheron, Joel E. Fischer, Stuart Reeves, and Sarah Sharples. Voice interfaces in everyday life. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 1–12, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3173574.3174214.

18 Alisha Pradhan, Kanika Mehta, and Leah Findlater. "Accessibility Came by Accident": Use of Voice-Controlled Intelligent Personal Assistants by People with Disabilities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 1–13, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3173574.3174033.

19 Parisa Rashidi and Alex Mihailidis. A Survey on Ambient-Assisted Living Tools for Older Adults. *IEEE Journal of Biomedical and Health Informatics*, 17(3):579–590, 2013. doi:10.1109/JBHI.2012.2234129.

20 Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8689–8696, April 2020. doi:10.1609/aaai.v34i05.6394.

21 Dingmin Wang, Chenghua Lin, Qi Liu, and Kam-Fai Wong. Fast and Scalable Dialogue State Tracking with Explicit Modular Decomposition. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–295, Online, 2021. Association for Computational Linguistics. doi:10.18653/v1/2021.naacl-main.27.

22 Inês Águia, Pedro Carneiro, Rafael Pinto, Ana Patrícia Rocha, Nuno Almeida, and António Teixeira. Integrating Conversational Assistants with Smart Mirrors for Mental Health Assessment. In *Companion of the 2025 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp Companion '25, pages 1202–1208, New York, NY, USA, 2026. Association for Computing Machinery. doi:10.1145/3714394.3756237.